Encoding of Multiple Depth Streams

Sang-Uok Kum

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2008

Approved by:

Ketan Mayer-Patel, Advisor

Marc Pollefeys, Committee Member

Leonard McMillan, Committee Member

Henry Fuchs, Committee Member

Anselmo Lastra, Committee Member

ABSTRACT

Sang-Uok Kum: Encoding of Multiple Depth Streams.
(Under the direction of Ketan Mayer-Patel.)

With advances in hardware, interests in systems for capturing, representing, and transmitting dynamic real world scenes have been growing. Examples of such systems include 3D immersive systems, tele-immersion video conferencing systems, 3D TV, and medical consultation and surgical training.

These systems use multiple cameras for dynamic scene acquisition. One approach for scene acquisition is to use multiple video streams captured from varying viewpoints in order to form a dynamic light field and then to use image based rendering (IBR) techniques for generating virtual views of the scene. Another approach is to use the imagery from the multiple cameras to derive a 3D representation of the environment that is then transmitted with color.

The dynamic light field approach handles complex scenes where 3D reconstruction would be difficult. Additionally, the captured video streams can be transmitted with little further processing, while deriving 3D information requires extensive processing before transmission. However, more cameras are needed for dynamic light fields since light fields need to be acquired more densely to be as effective.

One common 3D representation for the dynamic environments is multiple depth streams – streams of images with color and depth per pixel. A large number of depth streams are needed to properly represent a dynamic scene. Without compression, multiple depth streams, even sparsely sampled, requires significant storage and transmission bandwidth. Fortunately, the strong spatial coherence between streams and temporal coherence between frames allows for an effective encoding of multiple depth streams. In this dissertation, I present an effective encoding algorithm for multiple depth streams that uses approximate per pixel depth information to predict color for each pixel.

ACKNOWLEDGMENTS

First, I would like to thank my advisor Ketan Mayer-Patel for guiding me through my work, helping me overcome major hurdles, and challenging me with critical feedback. He has been a great mentor. Also thanks to my dissertation committee members, Marc Pollefeys, Leonard McMillan, Henry Fuchs, and Anselmo Lastra for their helpful advice and suggestions.

I appreciate all the support, advice, and feedback from Herman Towles and the members of the Office of The Future. Thanks to Hye-Chung Kum, David Gotz, Scott Larsen, Jason Repko, and Sudipta Sinha for sharing their ideas and code. I also thank all of the UNC-CS graduate students that I have known for making my years in Chapel Hill enjoyable, and UNC-CS staff for the technical and administrative support I have received over the years.

I thank all the XviD developers and contributors for sharing their code which has been invaluable to my work. Also thanks to Larry Zitnick, Sing Bing Kang, and Microsoft Research for sharing the 3D Video data that have been used in my dissertation.

I would like to thank the funding agencies – Link Foundation for the generous fellowship, National Science Foundation (grants ANI-0219780 and IIS-0121293), and Advanced Network and Services, Inc.

Thank you my parents for always believing in me, being there, and making me who I am. Also thanks to my grandparents, who encouraged me to be ambitious, taught me to never lose hope, and always reminded me that if there is a will there is a way.

Finally, to my wife, Byoung-Hwa, and my three wonderful children, thank you for your love, support, and encouragement. This journey would not have been possible without you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

3DAV	3D Audio and Video
dB	decibel
DCT	discrete cosine transform
DWT	discrete wavelet transform
fps	frames per second
GlSAS	global squared angle sum
GSAS	group squared angle sum
IBR	image based rendering
IBVH	image-based visual hull
ICT	irreversible component transformation
ISO	International Organization for Standardization
JPEG	Joint Photographic Experts Group
LDI	layered depth image
LSAS	local squared angle sum
MPEG	Moving Picture Experts Group
PSNR	peak signal-to-noise ratio
RCT	irreversible component transformation
RGB	red, green, and blue
RLE	run-length encoding
ROI	region of interest
SDLF	sparse dynamic light fields
TSAS	total squared angle sum
VQ	vector quantization
YCbCr	luminance, blue and red chrominance

Chapter 1

Introduction

With recent advances in hardware, the interest in real-time interaction with real world environments has increased. Using multiple cameras and computers, a dynamic real world scene can be captured, encoded, transmitted, and rendered in real-time. These advances has motivated development of real-time interactive systems for 3D immersion [Gross et al. 2003], tele-immersion video conferencing [Kauff and Schreer 2002, Towles et al. 2002, Baker et al. 2002], 3D TV [Fehn et al. 2002, Matusik and Pfister 2004], and medical consultation and surgical training [Welch et al. 2005].

These real-time interactive 3D systems all have three main components in common – acquisition, scene encoding and transmission, and rendering. The acquisition system consists of multiple cameras and controllers. The controllers ensures that the multiple cameras are synchronized and capture the environment simultaneously. After capturing the real world scene, the images must be encoded for transmission. The encoding should retain the 3D information of the scene to enable correct rendering of the environment from different views. Additionally, the encoding should be efficient enough for transmission of the data to the remote site in real-time. The transmitted data is then rendered at interactive rates by the rendering system. The rendering system also tracks the user to interactively generate the correct view of the environment from the user's viewpoint.

The data representation of the 3D environment influences the design of the real-time interactive 3D systems since it affects encoding and rendering. One method to represent the 3D environment is to reconstruct the geometry of objects in the scene using the images from multiple

views. One such model is the image-based visual hull (IBVH) [Matusik et al. 2000] used in the Coliseum [Baker et al. 2002]. IBVH projects the object silhouettes from different camera views and computes the intersection to generate a 3D model. Another model that has been used is the video fragments [Würmlin et al. 2004] for the blue-c system [Gross et al. 2003]. Video fragments are a point based representation which exploits spatio-temporal coherence by identifying differential fragments from multiple views in 2D image space. The 3D point representation of the scene is updated with these differential fragments.

Another approach is to use multiple video streams to form a dynamic light field and use image based rendering (IBR) techniques to generate virtual views of the environment [Matusik and Pfister 2004]. A light field represents a static 3D scene by modeling the light rays with a set of 2D images [Levoy and Hanrahan 1996]. Virtual views can be generated using light fields by sampling light rays of the scene from the desired view. A dynamic light field is a stream of light fields. Dynamic light fields have the advantage of being able to handle complex scenes where 3D reconstruction of objects would be difficult. Additionally, the captured videos can be transmitted with little further processing. However, one of the drawbacks is that the environment needs to be acquired more densely, i.e. more views are needed, than the geometry model to be effective [Chai et al. 2000].

*Depth stream*s combine the above two approaches. A depth stream is a stream of frames with color and depth value per pixel. Since depth streams contain 3D geometry information directly for each pixel, these streams do not need to be acquired as densely as light fields. Also, depths streams can be used to acquire complex scenes just as light fields are used. Depth streams have been used in a number of real-time interactive 3D systems [Kauff and Schreer 2002, Towles et al. 2002, Fehn et al. 2002].

In this dissertation, I examine the encoding aspects of multiple depth streams. Specifically, I propose an algorithm for effectively encoding multiple depth streams using spatial coherence between streams and temporal coherence between frames.

(a) Color **(b) Depth**

Figure 1.1: Depth Image

1.1 Multiple Depth Streams

A *depth image* represents a static 3D scene as an image with color and depth information per pixel [McMillan and Bishop 1995]. Depth images can be used to render a scene from different views by 3D warping: An image from a desired view can be generated by projecting the pixels from the depth image into the actual 3D locations in the real world, and then reprojecting them onto the desired view. An example depth image from [Zitnick et al. 2004] is shown in Figure 1.1. A depth image can be generated in real-time using computer vision algorithms [Kanade et al. 1996, Schreer et al. 2001, Mulligan et al. 2002, Yang and Pollefeys 2003]. There are also efforts to develop hardware that can acquire depth images directly [Canesta, ZCam].

One major drawback with using a single depth image is the presence of disocclusion artifacts. Disocclusion artifacts are caused when a portion of the scene not visible in the original depth image is visible from the desired view. Using multiple depth images from multiple views can reduce these disocclusion artifacts. Layered depth images (LDI) merge multiple depth images into a single depth image by 3D warping the multiple depth images into a single view and keeping multiple depth values per pixel [Shade et al. 1998]. However, the fixed resolution of an LDI imposes limits on the sampling of the multiple depth images, and may not provide adequate sampling rates for every depth image. An LDI tree, an octree with a single LDI in each node, was proposed to overcome this limitation [Chang et al. 1999]. The LDI tree preserves the sampling rates of the reference images by using a hierarchical data structure.

A *depth stream* is a stream of depth images which can be used to represent a dynamic 3D environment. Generally, multiple depth streams are used to represent a dynamic 3D scene

to minimize disocclusion artifacts. Multiple depth streams exhibit strong spatial coherence between streams since they capture the same environment from different views. Additionally, they exhibit temporal coherence between frames since motion in the scene between frames are usually small. These two characteristics can be used to effectively encode multiple depth streams.

The initial step in encoding multiple depth streams is to select *reference streams*. The reference streams serve as the basis streams for spatial prediction by the *non-reference streams*. Therefore, the reference streams must be encoded independently of other streams, and only utilize the temporal coherence between frames. Once reference streams have been selected and encoded, non-reference streams can be encoded using reference streams for spatial prediction. The depth value associated with each pixel in the non-reference stream can be used to project that pixel into the corresponding frame of the reference stream. If the color of the corresponding pixel in the reference stream is similar to the color of the projected pixel of the non-reference stream, the color information for that pixel of the non-reference stream no longer needs to be encoded. Instead, simply encoding the depth and indicating that the color can be derived from the reference stream is sufficient.

In this dissertation, I examine different approaches for selecting reference streams for multiple depth stream encoding. I also investigate methods for extending traditional video encoding techniques to encode reference streams. Specifically, I examine how to apply *motion compensation*, a video encoding technique for utilizing temporal coherence between frames, to encode depth streams – frames with color and depth. Finally, I propose an algorithm to encode non-reference streams utilizing reference streams to spatially predict reference colors per pixel using depth.

1.2 Thesis Statement

The thesis of my dissertation is the following:

Multiple depth streams can be effectively encoded using approximate per pixel depth to predict a color at each pixel without the use of motion compensation. This encoding is comparable to encoding these same streams using traditional video encoding techniques that employ

motion compensation.

1.3 Contributions

The contributions of this dissertation include:

- Identifying different approaches for selecting reference streams,
- Developing evaluation methods for reference stream selection, and employing them to evaluate the reference stream selection algorithms,
- Investigating methods for encoding reference streams by extending traditional video encoding techniques to encode depth streams,
- Introducing a new methodology for evaluating motion compensation, and applying it to evaluate reference stream encoding,
- Examining methods for encoding depth values of depth streams, independent of color,
- Describing a novel algorithm for encoding color values of non-reference streams using spatial and temporal prediction per pixel without motion compensation,
- Evaluating the non-reference stream encoding algorithm with traditional video algorithms.

1.4 Organization

The rest of the dissertation is organized as follows. Chapter 2 reviews related works for multiple depth stream encoding – image encoding of JPEG and JPEG2000, video encoding, and multi-view coding. Chapter 3 describes how to extend traditional video encoding techniques to encode a depth stream. Chapter 4 details an algorithm to encode depth streams using spatial coherence between streams and temporal coherence between frames. Chapter 5 examines how to select reference streams to use as a basis stream for spatial prediction when encoding multiple depth streams. Chapter 6 describes multiple depth stream encoding. Finally, in Chapter 7, I

conclude the dissertation with a summary of the research and present possible areas for future work.

Chapter 2

Related Works

In this chapter, an overview of related work for multiple depth stream encoding is presented; image encoding, video encoding, and multi-view coding. First, in Section 2.1 an overview of JPEG and JPEG2000 image encoding techniques are described. Then video encoding, which extends image encoding by incorporating temporal coherence between frames, is presented in Section 2.2. Finally, Section 2.3 describes multi-view coding of static and dynamic scenes which utilize spatial coherence between views and temporal coherence between frames.

2.1 Image Encoding

In this section, the image compression algorithms relevant to the dissertation – JPEG and JPEG2000 – are summarized.

2.1.1 JPEG

JPEG [JPEG 1994] is the image encoding standard from ISO/IEC Joint Photographic Experts Group (JPEG). In this section, a subset of the JPEG standard called baseline JPEG is described. Baseline JPEG uses the discrete cosine transform (DCT) and Huffman encoding with 8-bit precision per image component.

A color image is generally represented as an array of pixels. Pixel color is defined using the three primary colors – red, green, and blue (RGB). However, when encoding a color image, the RGB components are usually transformed into a different color space that better represents the human visual system – luminance (Y) and two chrominance (Cb and Cr) components.

Due to the densities of color and brightness sensitive receptors in the human eye, humans can see considerably more fine detail in the brightness of an image (Y) than in the color of an image (Cb and Cr). Therefore, when encoding an image represented in YCbCr color space, the chrominance components are generally subsampled in half resolution vertically and horizontally for higher compression with little effect on perceived quality of the image.

The basic unit of JPEG encoding is a *block* – 8×8 array of pixels. Each component is divided into 8×8 blocks, and each block is encoded individually by the following steps:

1. 2D DCT,
2. Quantization,
3. Entropy encoding.

The DCT converts each block into the spatial frequency domain to better represent the spatial coherence of neighboring pixels. The 2D DCT takes an 8×8 image block, and generates an 8×8 array of DCT coefficients. The first coefficient is referred to as the DC coefficient since it is the coefficient for zero frequency, while all other coefficients are referred to as the AC coefficients. The DC coefficient represents the average value of the pixels in the block, and is generally significantly larger than the AC coefficients. In the spatial frequency domain, for a typical real world image, most of the large coefficients have a tendency to be low frequency coefficients, and very few are high frequency coefficients. Additionally, the human visual system is more sensitive to low frequency information than high frequency information. Therefore, high frequency coefficients can be compressed more aggressively with little effect on the human perception of image quality.

After the DCT, the coefficients are quantized. Each coefficient is divided by the corresponding quantizer value in the *quantizer matrix* and rounded to the nearest integer. The larger the corresponding quantizer value, the smaller the range of the resulting quantized coefficients – i.e. requiring fewer bits for encoding. The quantizer matrix is also based on human perception, so high frequency coefficients are generally associated with larger quantizer values. This often sets many of the high frequency coefficients to 0. Also, a separate quantizer matrix is used for the luminance component and the chrominance components due to their different

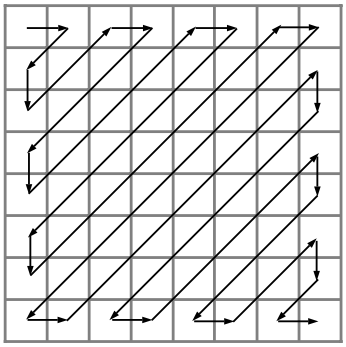


Figure 2.1: Zigzag Scanning

characteristics. The compression ratio of an image is controlled by the quantization factor: Higher quantizers result in higher compression but lower image quality.

Once quantization is finished, the DC coefficient and the AC coefficients are encoded separately. The DC coefficient is differentially encoded from the DC coefficient of the previous block since it is generally significantly larger than the AC coefficients and correlated from block to block. The 15 AC coefficients are ordered linearly by zigzag scanning (Figure 2.1). The linearly ordered coefficients are then encoded using run-length encoding (RLE): It is coded as a pair of *run-length* and *level*, where *run-length* defines the number of zero coefficients before a non-zero coefficient (*level*). The generated run-length pattern is then encoded using Huffman coding.

2.1.2 JPEG2000

JPEG2000 [JPEG 2000, Skodras et al. 2001, Taubman and Marcellin 2002] is a newer image encoding standard from the JPEG committee to succeed JPEG. JPEG2000 was created to address limitations of JPEG and is optimized for efficiency, scalability, and interoperability.

In JPEG2000, the original image is partitioned into non-overlapping blocks of equal size called *tiles* (only tiles at the image boundaries may be of different size). A tile is the basic encoding unit in JPEG2000 and each tile is encoded independently as if it was a separate image. The size of a tile can be arbitrary, and can be the same size as the image – i.e. one tile for the whole image. Each tile is encoded by the following steps:

1. DC level shifting,

2. Color component transformation,

3. 2D discrete wavelet transform (DWT),

4. Quantization,

5. Precinct and code block partition,

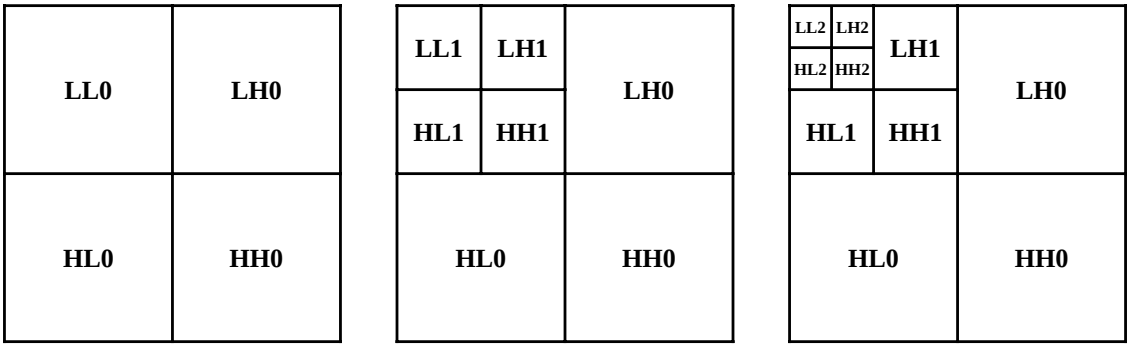
6. Entropy encoding,

7. Packet, layer, and code stream creation.

For each tile the first encoding step is DC level shifting. JPEG2000 supports both signed and unsigned images, but DC level shifting is only done for unsigned images. Every pixel in the unsigned image is subtracted by 2^{b-1} where b is the number of bits for the component. So for an 8 bit unsigned image, 2^7 is subtracted from every pixel.

The next step is the color component transformation. The color component transformation is applied only if the underlying image is a primary color image (i.e. an RGB image). As with JPEG, the objective of the transformation is to better represent the human visual system for more efficient encoding. Since, JPEG2000 supports both lossy and lossless encoding, there are two types of color component transformations. One is an irreversible component transformation (ICT) which can only be used for lossy encoding, and the other is a reversible component transformation (RCT) which can be used for both lossy and lossless encoding. ICT is very similar to the color transformation in JPEG and represents color using luminance (Y) and chrominance (Cb and Cr) components. However, the inverse ICT transformation does not result in the same RGB values since floating point arithmetic is used. Therefore, for lossless encoding, only RCT transform can be used. The RCT transform approximates the YCbCr transform using integer arithmetic to ensure the exact inverse transformation. Another difference from JPEG is that the chrominance components (Cb and Cr) is not subsampled to increased encoding efficiency. Instead, JPEG2000 uses the multi-resolution characteristics of the DWT by only using the LL subband to achieve the same effect.

After color component transformation, 2D DWT is used to decompose the tile into 4 subbands: LL, HL, LH, and HH. Each subband is a down-sampled lower resolution of the original,



(a) Level 1 **(b) Level 2** **(c) Level 3**

Figure 2.2: Discrete Wavelet Transform Decomposition

represented by coefficients. The LL subband is recursively decomposed until its size reaches a single pixel. An example of the first 3 levels of the DWT decomposition is shown in Figure 2.2. As with color component transformation, there are two types of DWT, irreversible and reversible. Any reversible DWT can be used for lossless encoding, and the default specified in JPEG2000 is the Le Gall 5-tap/3-tap filter [JPEG 2000]. Similarly, any irreversible DWT can be used for lossy encoding, and the default specified in JPEG2000 is the Daubechies 9-tap/7-tap filter [JPEG 2000]. Additionally, the irreversible DWT must be used if the ICT was used for color component transform, and the reversible DWT must be used for the RCT.

Once the tiles have been decomposed, all coefficients are quantized. Each subband is allowed only one quantizer, but the quantizer can be different between subbands. Since quantization reduces precision, it is a lossy transformation. Therefore, for lossless encoding, the quantizer must be 1 for all subbands.

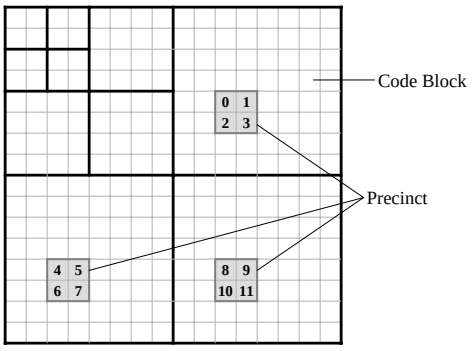


Figure 2.3: Tile Partitioned into Code Blocks and Precincts

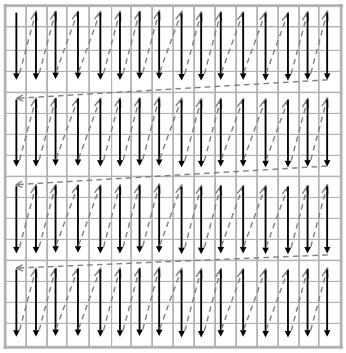


Figure 2.4: Code Block Bit Plane Scanning

Following quantization, each subband is divided into non-overlapping blocks. Corresponding blocks from each subband at each resolution level are grouped to create a *precinct*. Each precinct is further divided into non-overlapping blocks referred to as *code blocks*. The encoding order of the code blocks in a precinct is the raster scan order within the subband. Also, each code block is encoded independent of other code blocks. Figure 2.3 demonstrates an example tile divided into code blocks. One precinct is shaded, with the number indicating the scan order of the code blocks.

The code blocks are encoded on a bit plane basis, starting with the most significant bit. Each bit plane is scanned starting from top left, in units of 4 bit columns, in raster scan order (Figure 2.4). The scanned bits are encoded using a context-based adaptive binary arithmetic coder. Therefore, a code block is encoded into multiple bit streams.

Bit streams from each code block in a precinct is used to construct a *packet*. A packet contains information about a spatial location at a single resolution level. A *layer* is created from collection of packets, one from each precinct at each resolution level. Since a layer is a collection of packets, it contains information about the entire tile. Therefore, layers are used to progressively enhance image resolution, quality, spatial location, or components. The first layer is assembled to achieve the initial target, and subsequent layers are assembled to achieve successively higher targets. The sequentially ordered layers generate the final bit stream which is referred to as the *code stream*.

JPEG2000 offers the following features:

- Superior low bit-rate performance than JPEG.

- Multiple resolution encoding using multiple layers. The first layer is encoded to the initial spatial resolution. The subsequent layers successively increase the spatial resolution, and the final layer encodes the full spatial resolution of the image. Different spatial resolution image can be obtained by decoding up to the needed layers from a single code stream.

- Quality scalability. Each layer is encoded at full spatial resolution, but at a different quality. The first layer is encoded at the lowest target image quality, and successive layers are encoded at higher image quality with the final layer at highest target image quality. Only the layers that is required to achieve the desired image quality are selected from the code stream for decoding.

- Multiple target bit rate encoding. A code stream is generated by encoding each layer to achieve specified target bit rates at full spatial resolution. When a certain target bit rate is desired, only layers up to the specified target is required to be decoded.

- Improved error resilience. Each code block is independently encoded to minimize the impact of bit errors to a given code block. Also, after every coding pass, the arithmetic coder can be terminated and contexts reset.

- Region of interest (ROI) encoding. JPEG2000 allows a portion of the image to be encoded with better quality and less distortion than the rest of the image.

- Support of multiple color components, multiple bit depth (1-16 bits), and both signed and unsigned pixel values.

2.2 Video Encoding

Video is a stream of color images which are referred to as frames. Therefore, color image encoding techniques can be used to encode each frame. However, video has an additional characteristic not present in color images that can improve encoding – temporal coherence between frames. Generally, for a given sequence of frames, changes between consecutive frames are small. By encoding only the changes from frame to frame, video can be encoded more efficiently than encoding each frame individually.

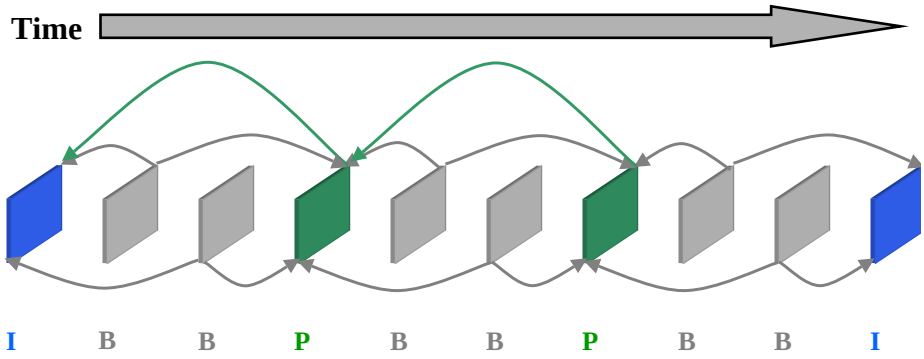


Figure 2.5: Video Sequence

There are many video encoding standards that have been developed. Some of the standards that are in wide use today are MPEG-2 [MPEG 1992], H.263 [ITU-T 1995], MPEG-4 [MPEG 2004], and AVC [MPEG 2005] (also referred to as H.264 or MPEG-4 Part 10). However, all video encoding standards are based on common general principles. Therefore, in this section, the general principles of video encoding are discussed instead of a particular standard. For more details on a specific standard refer to [MPEG 1992, Haskell et al. 1996, ITU-T 1995, MPEG 2004, MPEG 2005, Richardson 2003].

When encoding video, each frame is encoded either as an intra-frame or an inter-frame. An intra-frame, also referred to as an I-Frame, are encoded independently of other frames in order to allow decoding without requiring information from other frames. Therefore, the encoding of an intra-frame is very similar to JPEG encoding of color images.

An inter-frame, on the other hand, is dependent on one or more reference frames. For inter-frames, pixel values are predicted from these reference frames. The predicted values are then subtracted from the actual values to calculate the *residuals* which are encoded instead of the actual values. There are two type of inter-frames. One is the P-Frame which only uses a previous frame as a reference frame. The other is the B-Frame which use both previous and future frames as reference frames. Another difference between the P-Frame and the B-Frame is that a P-Frame can be used as a reference frame while a B-Frame cannot. Figure 2.5 shows an example of a video sequence with intra-frames and inter-frames. The arrows indicate the reference frames for the given frame.

All frames, intra-frames and inter-frames, start with converting the color into luminance (Y)

and chrominance (Cb and Cr) components. As with JPEG, the conversion is done to better match the human visual system. Additionally, the chrominance channels are usually subsampled to half resolution vertically and horizontally for better compression.

After the color space conversion, each component is partitioned into blocks (most video standards use a size of 8×8 pixels). Each block is encoded either as an intra-block or an inter-block. An intra-block is encoded independent of other frames, and encoded very similar to JPEG blocks. It is encoded by first doing a discrete cosine transform (DCT), and then the coefficients are quantized using a quantization matrix. Finally, the DC coefficient is differentially encoded among the intra-blocks of the frame, while the AC coefficients are variable length encoded using run length encoding and Huffman encoding. Some standards also allow the use of arthimetic coding instead of Huffman coding. Since intra-frames must be decoded independent of other frames, all intra-frame blocks are encoded as intra-blocks.

An inter-block is encoded with a *motion vector* and residuals. The motion vector specifies a block of equal size in the reference frame, which is used for prediction. The residuals are calculated by subtracting the predicted values from the actual values of the inter-block. This is referred to as *motion compensation*. For motion compensation to be effective, the predicted values should closely match the inter-block such that the calculated residuals are small. Also, to minimize the overhead of searching for motion vectors, motion compensation is applied on a *macroblock* basis. A macroblock corresponds to a block size of 16×16 pixels; usually it consists of four Y blocks, one Cb block, and one Cr block. When motion compensation is effective, i.e. the residuals are small, the overhead of encoding the motion vector with the residuals is less than encoding the block as an intra-block.

The motion vector is differentially encoded from prior motion vectors, and the difference is encoded using variable length codes. Additionally, the residuals are encoded in similar steps as the intra-blocks – DCT, quantization, and variable length encoding. The main difference being the quantization matrix. In intra-blocks, each element in the quantization matrix has a different value based on the characteristics of human perception. For inter-blocks, since the values are residuals, the default quantization matrix has the same value for all elements. Inter-frames are encoded using inter-blocks as well as intra-blocks. Intra-blocks are used when a suitable prediction cannot be found in the reference frame and encoding the block as an

inter-block becomes more expensive.

Intra-frames are encoded at higher quality than inter-frames because they are used as reference frames. Additionally, since intra-frames can be decoded independently of other frames, frames are encoded as intra-frames at certain intervals in the video to allow random access and faciliate seeking. For inter-frames, P-Frames are encoded at higher quality than B-Frames since P-Frames can subsequently be used as reference frames. Finally, the bitrate of a video is controlled by dynamically changing the quantizer values. As with image compression, larger quantizers yield greater compression and lower video quality.

2.3 Multi-view Coding

With advances made to digital cameras, it has become easier to capture a scene using multiple cameras from various view points. Generating images from multiple views allows better greater user interaction with the scene, since it can be rendered from various view points. In Section 2.3.1, multi-view coding of static scenes are discussed, while multi-view coding of dynamic scenes are presented in Section 2.3.2.

2.3.1 Static Scenes

Multi-view coding of static scenes have been widely researched in the field of computer graphics as image based rendering (IBR). Unlike traditional computer graphics, IBR uses images rather than geometry to render novel views. Light field rendering, one of the IBR techniques, represents a static 3D scene by modeling its light rays with a set of 2D images ([Levoy and Hanrahan 1996]). In [Levoy and Hanrahan 1996], the light field images were compressed using vector quantization (VQ). Improvements over VQ encoding was made using methods based on image compression; discrete cosine transform (DCT) in [Miller et al. 1998], and discrete wavelet transform (DWT) in [Lalonde and Fournier 1999, Peter and Straßer 2001]. The improvements were due to utilization of local spatial coherence in the images by transforming the images into the spatial frequency domain.

The light field acquires the static scene from multiple view points. Therefore, the light field images exhibit very high spatial coherence among different views. In [Magnor and Girod 2000],

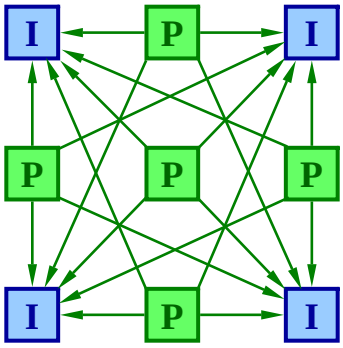


Figure 2.6: Light Fields with Disparity Compensation

disparity compensation was used to incorporate the spatial coherence between the images and improved encoding of the light field images. The light field images were first divided into I-Images and P-Images. The I-Images were encoded just like a normal color image with disparities. Using the disparities, P-Images predicted reference colors from I-Images and the difference between the reference colors and the actual colors were encoded using DCT. The reference structure of the frames are shown in Figure 2.6 where the arrows indicate the reference frame for the given frame.

2.3.2 Dynamic Scenes

Similar to static scenes, multi-view coding of dynamic scenes exhibit spatial coherence between views. However, they also have temporal coherence just like regular video. In [Matusik and Pfister 2004], multiple cameras were used to generate video streams of a dynamic scene from different viewpoints. Each video was encoded and streamed separately, i.e. only temporal coherence between frames was used for encoding. The spatial coherence between video streams were not used because of the complexity and the delay involved.

Multiple depth streams used for tele-immersion [Towles et al. 2002] were compressed by removing points redundant in the reference depth streams from the non-reference depth streams [Kum et al. 2003]. While this approach is scalable in the number of depth streams, the performance on real world data sets is not practical due to imperfect depth values
[Kum and Mayer-Patel 2005]. Also, this technique only takes advantage of spatial coherence between different streams but not the temporal coherence between frames of the same stream.

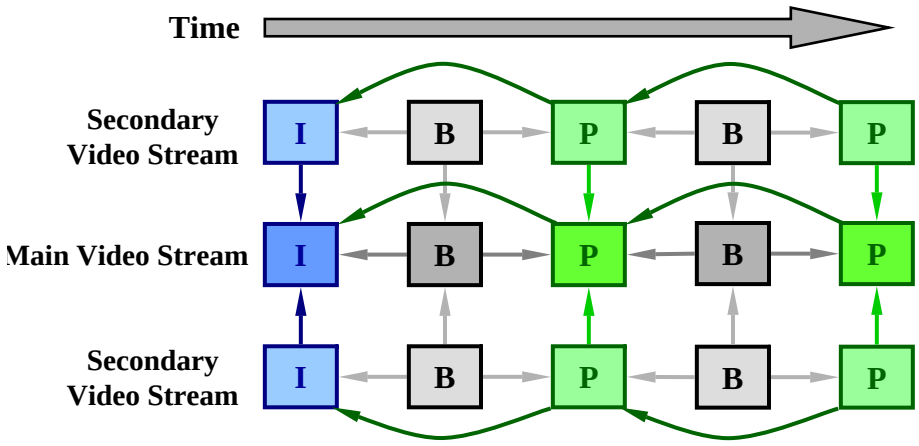


Figure 2.7: Sparse Dynamic Light Fields

Multiple video streams were compressed using both temporal and spatial coherence in Sparse Dynamic Light Fields (SDLF) [Chan et al. 2003]. Multiple video streams were categorized either as a main video stream or a secondary video stream. The main video stream was encoded in same manner as normal video. The secondary video stream used the correspond frame from the main video captured at the same time, as a reference frame in addition to the temporal reference frames. Figure 2.7 shows the reference frame structure for the SDLF where the arrows indicate the reference frames for the given frame.

Spatial and temporal coherence were used to compress multiple depth streams for 3D video in [Zitnick et al. 2004]. Temporal prediction was used to encode the reference streams, and spatial prediction was used to encode the non-reference streams. However, no frame used both temporal and spatial prediction.

3D video fragments were used in the blue-c system to encode multiple video streams [Würmlin et al. 2004]. 3D video fragments are a point based representations for dynamic scenes. It exploits spatio-temporal coherence by identifying differential fragments in 2D image space and differentially updating the 3D point representation of the scene.

Finally, the Moving Picture Experts Group (MPEG) set up an Ad-hoc Group on 3D Audio and Video (3DAV) [Smolic and Kauff 2005] due to rising interest in multi-view coding. The group has developed a draft for multi-view coding standard which uses both temporal and spatial reference frames for encoding [Vetro et al. 2006]. Figure 2.8 shows the proposed

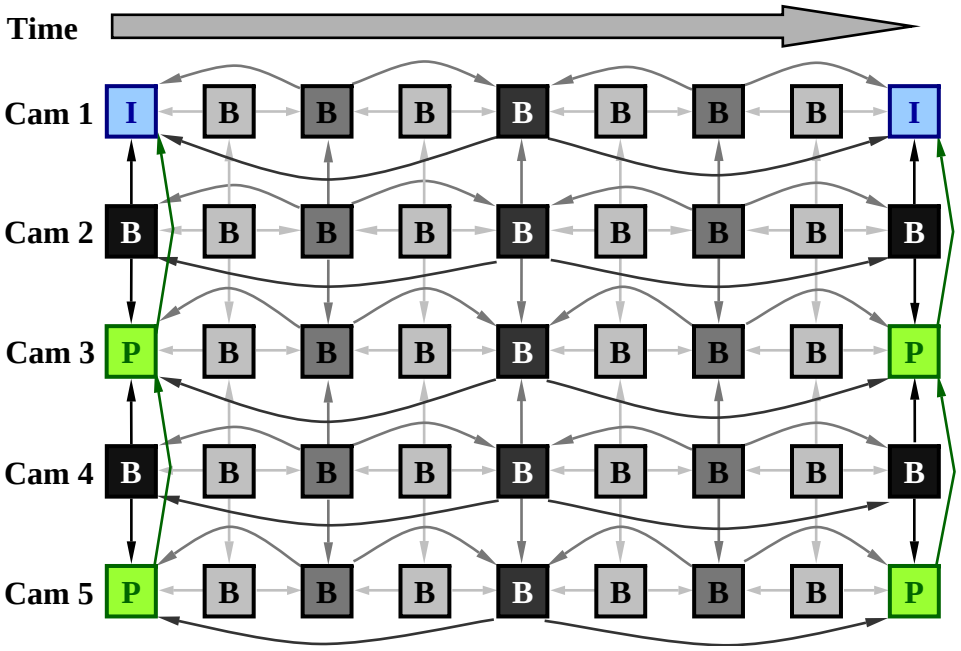


Figure 2.8: MPEG Multi-view Coding Standard

reference structure for multi-view coding, with the arrows indicating the reference frames for the given frame. The proposal allows a B-Frame (e.g. first frame of Cam 2 in Figure 2.8) to predict from a spatial reference frame taken at the same time – an I-Frame and a P-Frame. However, the P-Frame (first frame of Cam 3 in Figure 2.8) can also spatially predict from an I-Frame taken at the same time (first frame of Cam 1 in Figure 2.8). This increases delay when decoding, since total of three frames from the same point in time needs to be decoded.

Chapter 3

Intra-Stream

The initial step in compressing multiple depth streams – the encoding of intra-streams – is examined in this chapter. Multiple depth streams exhibit spatial coherence between streams as well as temporal coherence between frames within a stream. For effective encoding of multiple depth streams, these characteristics should be exploited. However, in order to utilize spatial coherence between streams, a *reference stream* – a stream which other streams refer to for spatial prediction – must be encoded such that it can be decoded independently of any other stream. Therefore, the reference stream can only use temporal coherence between its frames for encoding. Streams encoded only using temporal coherence, such as a reference stream, are referred to as I-Streams. An I-Stream is encoded by extending existing single stream video encoding techniques which make use of motion compensation. Depth is added as a fourth plane of information to the existing three plane representation for color. This chapter examines how well such a scheme works and investigates several ways of extending motion compensation to the new depth plane.

3.1 Encoding

A depth stream is very similar to a video stream in that both streams have three channels of color to encode. They both also have temporal coherence between frames. Therefore, existing video stream encoding techniques should be well suited for adoption in I-Stream encoding. The three color channels of I-Stream are encoded in a manner similar to standard video encoding, and depth is encoded as an 8 bit grayscale image.

(a) **Breakdancers Color**

(b) **Breakdancers Depth**

(c) **Ballet Color**

(d) **Ballet Depth**

Figure 3.1: Frame from Breakdancers and Ballet

3.1.1 Frame Encoding

Each frame of a depth stream consists of color and depth (Figure 3.1). The three components of color are encoded in YCbCr color-space – luminance and two chrominance components. Furthermore, the two chrominance components are subsampled by a factor of two both horizontally and vertically. Depth is encoded as an inverse value since the inverse of depth is linear in perspective space. However, using the inverse requires knowing the minimum and the maximum depth values which in turn affects the resolution of the depth values (Equation 3.1).

$$Depth_{(x,y)} = \frac{1}{\frac{Depth\ Inverse_{(x,y)}}{255} \times (\frac{1}{min} - \frac{1}{max}) + \frac{1}{max}} \tag{3.1}$$

In the scheme I have developed, each frame is encoded either as an I-Frame, or a P-Frame. An I-Frame is encoded without reference to other frames and can be decoded independently. The P-Frame uses temporal prediction from some prior reference frame. Unlike video encoding, no B-Frames are used because B-Frames depend on frames in the future which increases latency.

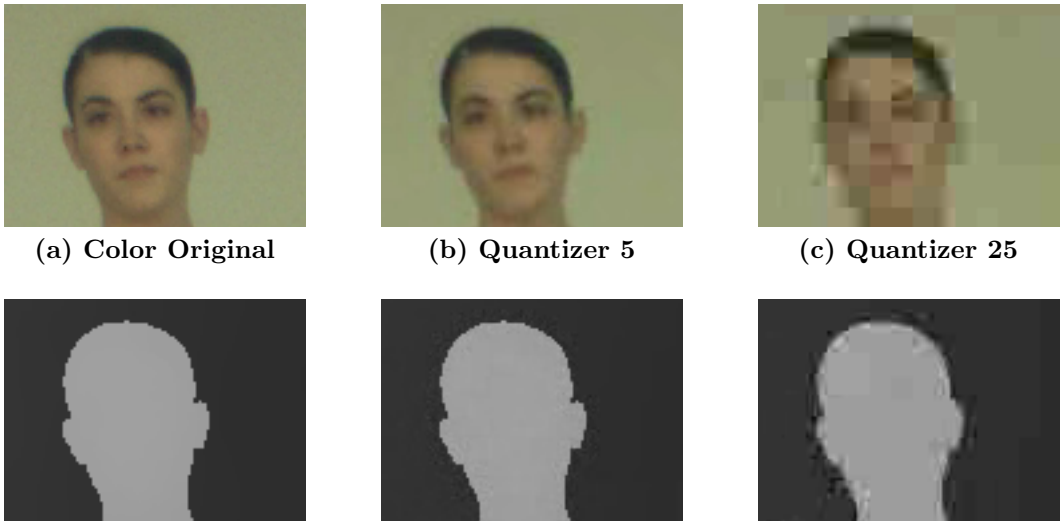
Each frame is encoded on a per block basis where the block size is 8×8 pixels and specified separately for each component.

3.1.2 Block Encoding

All blocks are encoded either as an intra-block or a inter-block. An intra-block is encoded by transforming the values with a 2D discrete cosine transform (DCT) and quantizing the DCT coefficients by a scale factor. After quantization, the DC term coefficient is differentially encoded from the previous block's DC term. The AC term coefficients are run length encoded with a pair of numbers – *run* and *value*. The *run* indicates the number of zeros before the *value*. Finally, the run length pairs are encoded using the intra block Huffman table used for MPEG-4 [MPEG 2004]. For depth, the intra block Huffman table specified for the Y component is used. All blocks of an I-Frame are encoded as intra-blocks since an I-Frame must be decoded independently of other frames.

A inter-block incorporates temporal coherence between frames for encoding by predicting the block from the reference frame. Each inter-block is associated with a *motion vector*. The motion vector specifies the predicted values for the block to be encoded from the reference frame. The reference values can start from any location and do not have to be at a block boundary within the reference image. The *residual* is the difference between the values of the block to be encoded and the predicted values and is then encoded along with the motion vector. As with the intra-block, the residual values are transformed with a 2D DCT and the DCT coefficients are quantized by a scale factor (i.e. the *quantizer*). However, unlike the intra-block, the DC term is encoded in the same manner as the AC terms. The coefficients are zero run length encoded using the MPEG-4 inter block Huffman table.

The compression efficiency of a inter-block depends on the residual that needs to be encoded. Therefore, the closer the predicted block matches the block to be encoded, the higher the compression ratio. P-Frame blocks are encoded using inter-blocks except for the following two cases. One is when the block does not have high frequency content and encodes substantially better as an intra-block. The other is when no adequate motion vector can be found, in which case the block is encoded as an intra-block.



(a) Color Original (b) Quantizer 5 (c) Quantizer 25

(d) Depth Original (e) Quantizer 5 (f) Quantizer 25

Figure 3.2: Closeup of a Frame from Ballet

3.1.3 Encoding Parameters

Quantizer

Quantization of DCT coefficients during encoding affects image quality and compression ratio. Increasing the quantization step size reduces the range of coefficients that need to be represented and eliminates small coefficients by reducing them to zero. In effect, the larger the quantizer, the higher the compression ratio. However, larger quantizers lead to greater distortion in the encoded images. The images becomes increasingly blurry since high frequency content is filtered out. Image quality, the amount of distortion introduced relative to the original image, is measured using peak signal-to-noise ratio (PSNR). PSNR in decibels (dB) is calculated using Equation 3.2.

$$\text{PSNR} = 20 \log_{10} \frac{255}{\sqrt{Mean\ Squared\ Error}} \tag{3.2}$$

For traditional video streams, the three color components are encoded with the same quantizer. However, for I-Streams, depth and color should not share the same quantizer since they have different characteristics. Figure 3.2, a closeup of the Ballet frame in Figure 3.1, shows the effect of quantization on color and depth. With a low quantizer (Figure 3.2b and 3.2e), distortion from the original (Figure 3.2a and 3.2d) is not noticeable. However, with

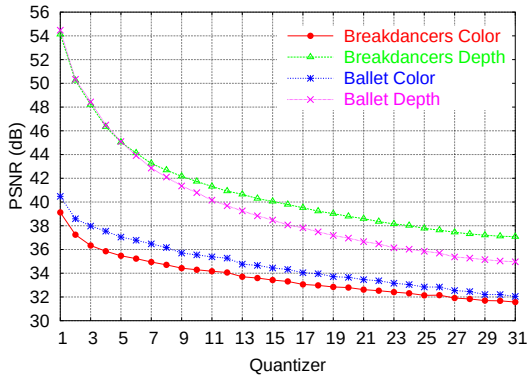


Figure 3.3: Effect of Quantizer on Image Quality

a high quantizer, the difference between color and depth is apparent. Generally, the effect of large quantizers is greater in color than in depth. Figure 3.2c shows Figure 3.2a encoded with quantizer of 25. The features of the face – eyes, eyebrows, nose, and lips – are barely distinguishable. Also block artifacts due to the block based encoding are evident. Figure 3.2f is Figure 3.2c encoded with quantizer of 25. There are noticeable artifacts around the boundary of the head but overall the images are similar. This is because depth values are usually smooth over a local region and only show high frequency content at object borders. So a higher quantizer can be used for depth than color to achieve the same PSNR.

Figure 3.3 shows the effect of quantization on image quality averaged over 100 frames for eight different streams. Color and depth is given for two separate data sets – Ballet and Breakdancers.

Motion Vectors

Inter-blocks must use extra bits to encode motion vectors compared to intra-blocks. However, the residuals for the inter-blocks will be very small if the predicted values match very well; for blocks that match very closely, there maybe little or no residual left to encode. This results in inter-blocks being encoded more efficiently than intra-blocks. Therefore finding a good motion vector is critical for effective encoding of inter-blocks. Video codecs generally use a motion vector based on the luminance (Y) plane for encoding all three color planes with good results. In other words, it is generally the case that plane-specific motion vectors for luminance and chrominance channels are not effective enough to justify the additional bits that

would be required to encode separate motion vectors for each plane. An important question for this work is whether the same is true for depth information. In the following discussion, we will refer to a motion vector derived from the Y plane as MV_Y and a motion vector derived from the depth plane as MV_D. Thus, for encoding inter-blocks of I-Streams, the following combinations are possible:

- Only use MV_D for encoding depth and color,
- Only use MV_Y for encoding depth and color, and
- Use MV_D to encode depth, and MV_Y to encode color.

Using only MV_D gives the optimal encoding for the depth values, at the expense of color value encoding. For example, there will be very few motion vectors that will be a good match for a block that contains the left eye in Figure 3.2a if the motion vector search is done in the Y plane. However that same block will result in a number of motion vectors that are a good match in depth (Figure 3.2d). Therefore, if MV_D is used to encode this block, the depth block will probably have very small residuals to encode, while the color blocks will most likely have to encode very large residuals.

Using a single motion vector based on the Y plane optimizes encoding of color values. The encoding efficiency for depth would depend on how well the motion vector reflects the real physical world. When searching for a good reference block, the actual real physical world motion of the block from the reference frame is not considered. Instead, only the closeness of the pixel color values of a block is considered. This sometimes results in motion vectors that do not reflect coherent motion in the real physical world, which in turn may result in a poor reference block for depth. An example would be a person with his arm in front of him in a long-sleeved shirt. The arm and the chest will have similar color but depth will be different. To encode a block where only the chest is present, MV_Y may select a reference block where the arm and the chest is both present if the color information is still a good match. However, this results in poor encoding for depth since it would have to encode the high frequency information resulting from the depth discontinuity between the arm and chest.

When using a single motion vector for encoding, whether its the MV_D or the MV_Y, the

total efficiency of I-Stream encoding depends on how well the specified motion vector can predict the other component. Using both MV_D and MV_Y for encoding will result in the best encoding for both color and depth values. However, this incurs the overhead of searching for and encoding an additional motion vector. Searching for a good motion vector is not a trivial task; a significant part of video encoding is spent on searching for motion vectors. Therefore the overhead of searching for an extra motion vector cannot be ignored.

3.2 Results

Two data sets from [Zitnick et al. 2004], Ballet and Breakdancers (Figure 3.1), were used to test I-Stream encoding. The I-Streams were encoded using a modified version of a well known video codec XviD [XviD]. XviD is an open source ISO MPEG-4 [MPEG 2004] compliant video codec. XviD was selected mostly because of source code availability, which can be modified without any restrictions. The modified XviD codec is able to encode frames with four components – RGB and depth – and the option to specify the motion vector used for encoding inter-blocks.

All frames were encoded as P-Frames with the previous frame always encoded as an I-Frame. This was done to eliminate error propagation when a P-Frame is encoded using a P-Frame as a reference. Also the quantizer for all blocks in a frame was kept constant for same components. This ensures that the encoding is affected similarly across blocks within the same frame, and a fair comparison can be made for luminance (Y) and depth.

3.2.1 Motion Vector

When encoding inter-blocks, as previously mentioned, two motion vectors can be considered. One is the *Y motion vector* (MV_Y) which is based on the luminance plane. The other is the *depth motion vector* (MV_D) based on the depth plane. If only one motion vector is used for encoding, the encoding efficiency will depend on how close the selected motion vector matches the non-selected one.

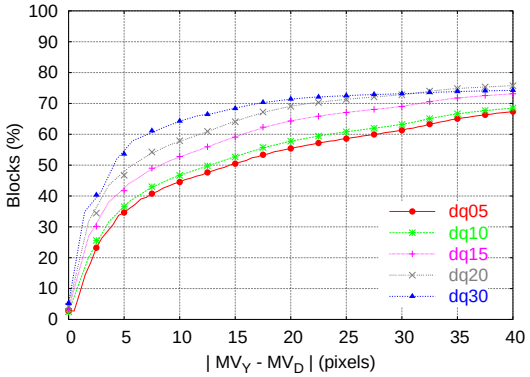
The *delta motion vector* (ΔMV), defined as $|MV_Y - MV_D|$, estimates how similar the two motion vectors match between corresponding blocks. If both motion vectors are a perfect

match, the size of the delta motion vector will be zero. As the vectors deviate, ΔMV will increase.

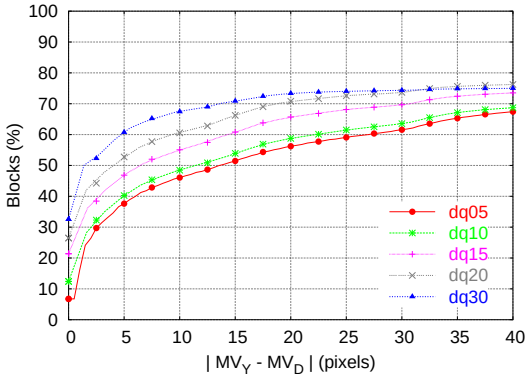
Figure 3.4 and Figure 3.5 shows the accumulative histogram of ΔMV for the Breakdancers data averaged over 8 streams of 99 frames each. For each figure, the X-axis represents ΔMV in pixels, and the Y-axis shows the percentage of inter-blocks that is less than or equal to the corresponding ΔMV. Figure 3.4 compares the ΔMV accumulative histograms at various depth quantizers – 5 (*dq05*), 10 (*dq10*), 15 (*dq15*, 30 (*dq30*) – for a given color quantizer; Figure 3.4a at color quantizer 5, Figure 3.4b at color quantizer 15, and Figure 3.4c at color quantizer 30. Figure 3.5 shows the accumulative histogram of ΔMV at different color quantizers – 5 (*cq05*), 10 (*cq10*), 15 (*cq15*), 30 (*cq30*) – for a given depth quantizer; Figure 3.5a at depth quantizer 5, Figure 3.5b at depth quantizer 15, and Figure 3.5c at depth quantizer 30. Similarly, Figure 3.6 and 3.7 is the ΔMV accumulative histogram of the Ballet data averaged over 8 streams of 99 frames each. Since there are no motion vectors defined for intra-blocks, all intra-blocks are ignored. This appears in the figure by the accumulative values not reaching 100% but plateauing at a lower point. The figures show that Breakdancers data has more intra-blocks than the Ballet data. This is primarily due to the fact that the Breakdancers sequence has faster motion between frames which results in less coherence between frames. It also shows that quantizer has little effect on the number of intra-blocks per frame.

As the quantizers are increased, the number of blocks with similar MV_Y and MV_D (i.e. smaller ΔMV) increase. When both the color and depth quantizer is at 30, most of the blocks have zero ΔMV; MV_Y and MV_D of the block are equal. This is because as the quantizer increases the high frequency content is reduced (Figure 3.2), resulting in a blurry image. Subsequently, as the image becomes more blurry, the feature differences in depth and color decrease, resulting in more similar motion vectors. In other words, as the quantizer becomes larger, the finer texture details in color are lost to the blurring effect, while coarser color discontinuities at the object boundaries remain. Since these boundaries are more likely to match the depth boundaries, the selected motion vectors are also more likely to match.

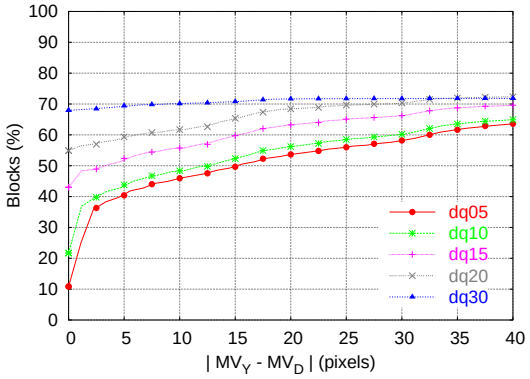
As the quantizer of color varies, ΔMV does not change significantly (Figure 3.5 and 3.7). However, altering the depth quantizer notably effects ΔMV (Figure 3.4 and 3.6). This is the result of the fact that the high-frequency content in a depth image is concentrated at object



(a) MV_Y at Quantizer 5

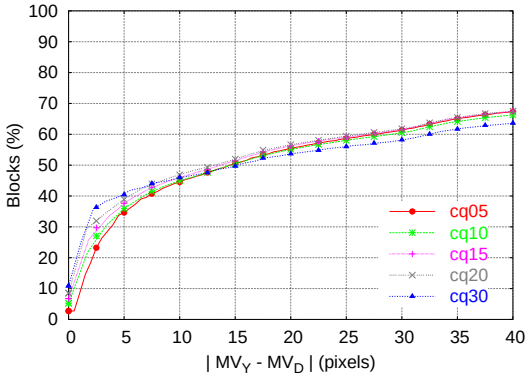


(b) MV_Y at Quantizer 15

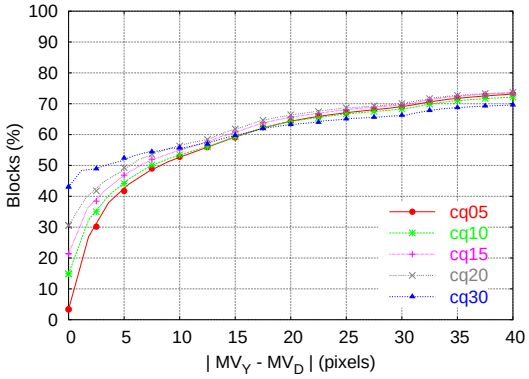


(c) MV_Y at Quantizer 30

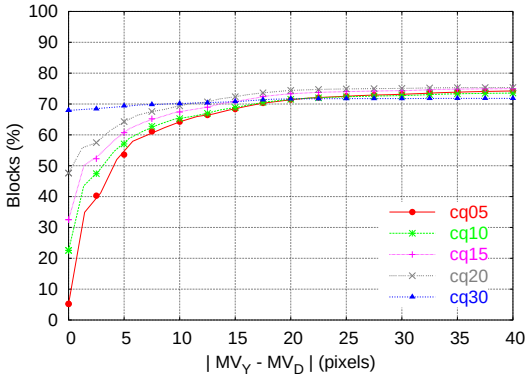
Figure 3.4: Motion Vector Difference Between MV_Y and MV_D for Breakdancers



(a) MV_D at Quantizer 5

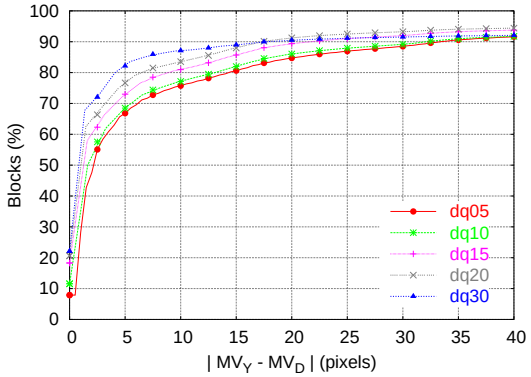


(b) MV_D at Quantizer 15

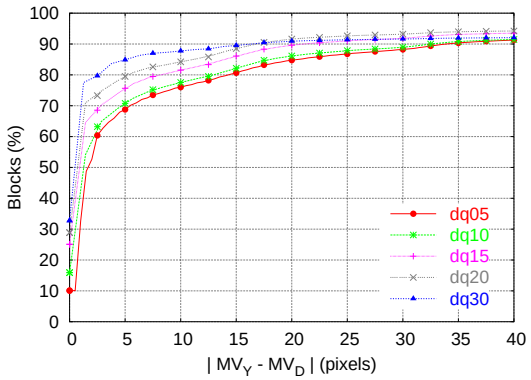


(c) MV_D at Quantizer 30

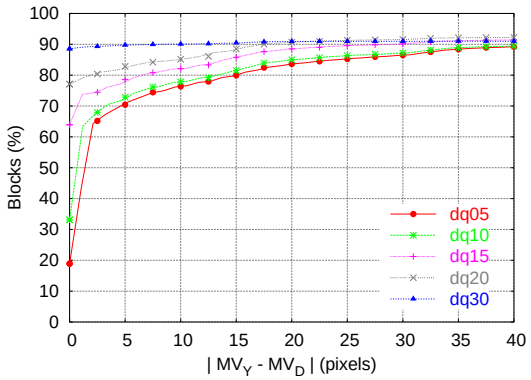
Figure 3.5: Motion Vector Difference Between MV_D and MV_Y for Breakdancers



(a) MV_Y at Quantizer 5

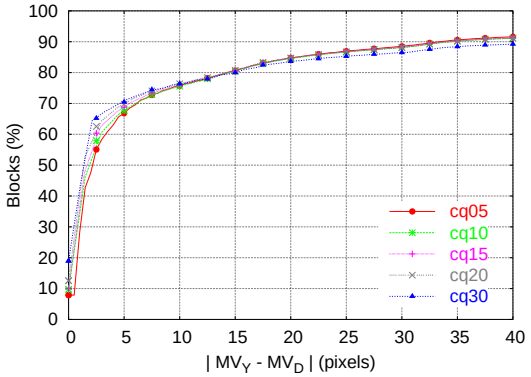


(b) MV_Y at Quantizer 15

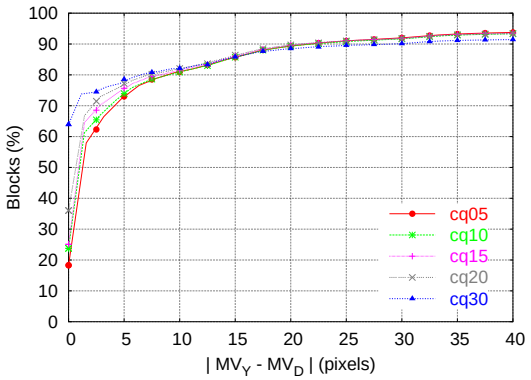


(c) MV_Y at Quantizer 30

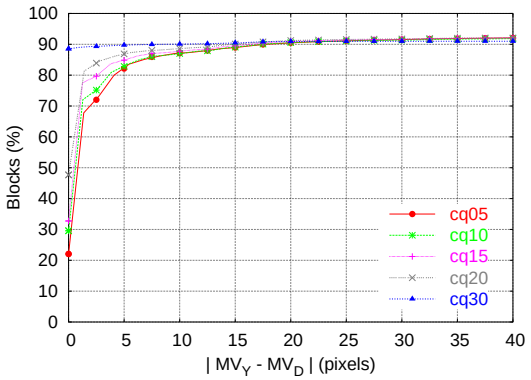
Figure 3.6: Motion Vector Difference Between MV_Y and MV_D for Ballet



(a) MV_D at Quantizer 5

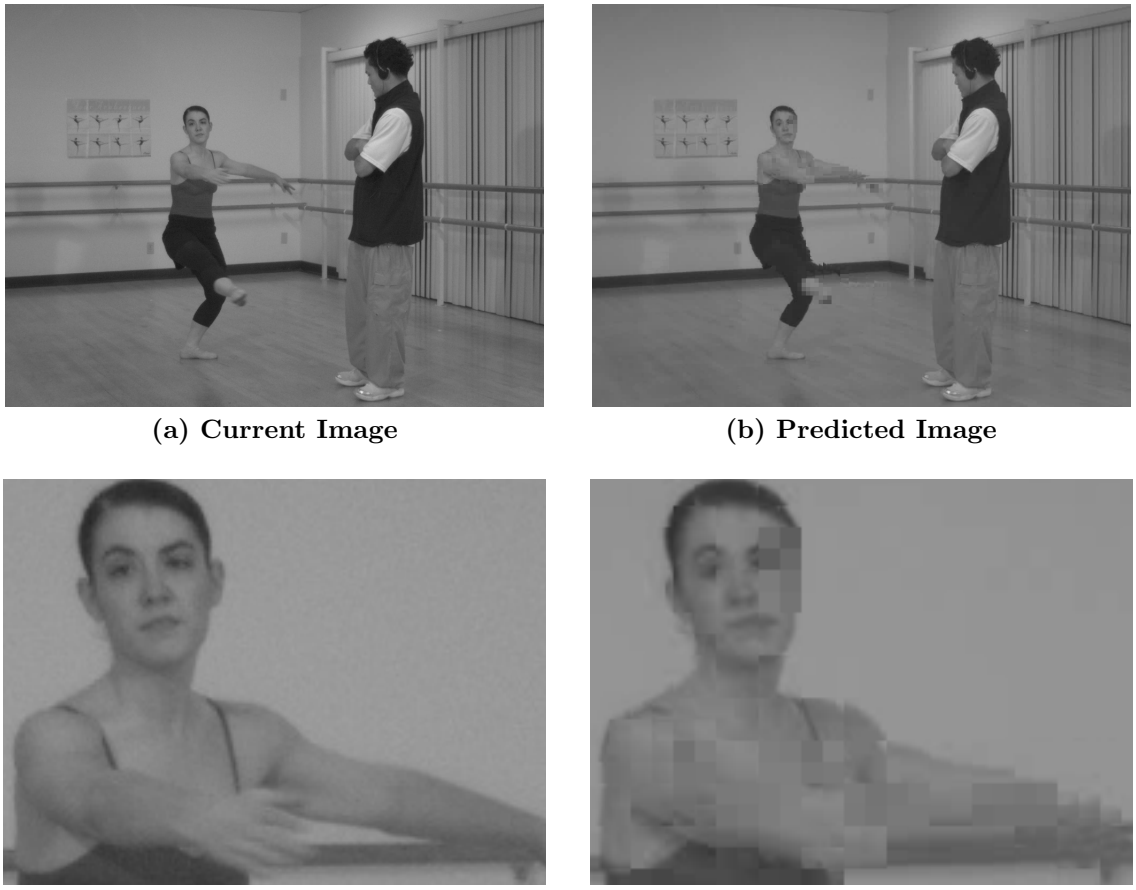


(b) MV_D at Quantizer 15



(c) MV_D at Quantizer 30

Figure 3.7: Motion Vector Difference Between MV_D and MV_Y for Ballet



(a) Current Image **(b) Predicted Image**

(c) Closeup of current Image **(d) Closeup of predicted Image**

Figure 3.8: Luminance Predicted Image of Ballet

boundaries. Consequently, for many blocks of depth which are wholly within an object, there are multiple reference blocks which are good choices. However, for color, due to the high frequency information spread throughout the image, there are fewer suitable reference blocks.

3.2.2 Predicted Image

In order to compare the three possible methods for motion compensation in P-Frames, the *predicted image* is generated for each frame to be compared. For inter-blocks, the pixel value of the predicted image is simply the predicted pixel values derived from motion compensation. For intra-blocks where no motion vector is present, there are no predicted values to use. There are two reasons for blocks being encoded as intra-blocks in P-Frames. One is that the block does not have high frequency content and encodes significantly better as an intra-block. The other is where no good motion vector can be found and encoding the block as an intra-block is

(a) Original Image **(b) Uniform Residual of 3** **(c) Residual of 30 for 10×10**

Figure 3.9: Predicted Images with Identical PSNR

more efficient. Ignoring intra-blocks, when generating a predicted image, would not adequately penalize the blocks where no good motion vector can be found. However, penalizing all intra-blocks equally, would not be reasonable for blocks where encoding it as an intra-block is inherently more efficient – when no high frequency content exists. Therefore, for intra-blocks, the predicted image for the block is set to the intra-block's DC value. For blocks with no high frequency content, using the DC component as the reference is a good estimation of the block; the residual between the block and the DC component is small. For blocks with no good motion vectors, the DC component, which represent the average intensity of the block, is a good representation of the block for the purposes of calculating the residual information that remains to be coded as non-DC coefficients.

Figure 3.8 shows an example of a luminance predicted image with its original frame. Figure 3.8a is the current image to be encoded, and Figure 3.8b is the predicted image generated from the reference image. Figures 3.8c and 3.8d is a closeup of Figures 3.8a and 3.8b. The intra-blocks, where DC value of the block has been used for the predicted image, can be seen in parts of the face and the arm in the predicted image (Figure 3.8d).

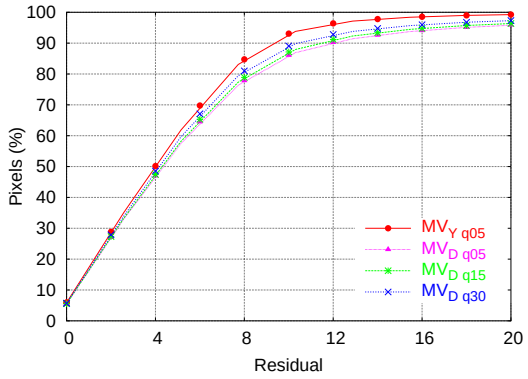
One metric used to compare the predicted images generated by the different motion compensation options is PSNR (Equation 3.2). PSNR measures the residual of the pixels in the current image (Figure 3.8a) from the predicted image (Figure 3.8b). The residual of a pixel is the difference between the current pixel value and the value of the pixel in the predicted image at the same location. Since PSNR measures the overall residual, it does not always accurately indicate how suitable the predicted image is for encoding the current image. An example is shown in Figure 3.9 where the image resolution of the original image (Figure 3.9a) is 100×100

pixels. Figure 3.9b is a predicted image which result in a uniform residual of 3 for all pixels. Figure 3.9c result in a uniform residual of 0 for all pixels except for a block of 10×10 pixels (near the right eye of the ballerina) which result in residual of 30 for each pixel. The PSNR for both of the predicted images will be the same since the mean squared residual is the same – $(3^2 \times 100 \times 100)/100^2 = 9$ for Figure 3.9b, and $(30^2 \times 10 \times 10)/100^2 = 9$ for Figure 3.9c. However, Figure 3.9b is a better predicted image for encoding since all pixels have a residual of 3, while the Figure 3.9c results in 100 pixels with the residual of 30.

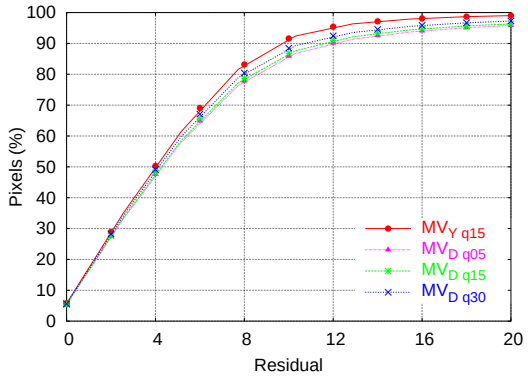
Therefore another metric – *accumulative residual histogram* – is also used to compare the predicted images generated by the different motion compensation options. An *accumulative residual histogram* shows the percent of pixels in the image that have equal or less than the given residual. Using the accumulative residual histogram, a predicted image which has more pixels with smaller residuals is considered better. So for the previous example of the two predicted images, the first predicted image would be considered better than the second predicted images because 100% of the pixels in the first predicted image has residual of 1, while in the second predicted image, 75% of pixels has a residual 0 but to reach 100% the residual becomes 10.

Accumulative Residual Histogram

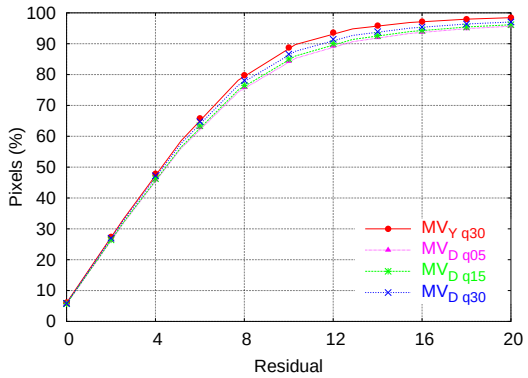
Figure 3.10 and 3.11 shows the average accumulative residual histogram for Breakdancers data averaged over 8 streams of 99 frames each. For each figure, the X-axis shows the residual, and the Y-axis indicates the percentage of pixels where the residual is less than or equal to the corresponding value. Figure 3.10 shows the luminance (Y) average accumulative residual histogram for luminance quantizers 5 (a), 15 (b), and 30 (c). The predicted images for each figure were created using MV_Y at specified quantizer, and MV_D at quantizers 5 (MV_{Dq05}), 15 (MV_{Dq15}), and 30 (MV_{Dq35}). Figure 3.11 shows the average accumulative residual histogram of depth at depth quantizers of 5(a), 15(b), and 30(c). The predicted images at each figure were generated using MV_D at specified quantizer, and MV_Y at quantizers 5 (MV_{Yq05}), 15 (MV_{Yq15}), and 30 (MV_{Yq30}). Similarly, Figure 3.12 shows the average accumulative residual histogram of luminance (Y) for Ballet data at Y quantizers of 5 (MV_{Yq05}), 15 (MV_{Yq15}), and 30 (MV_{Yq30}), and Figure 3.13 shows the average accumulative residual histogram of depth for Ballet data at depth quantizers of 5 (MV_{Dq05}), 15 (MV_{Dq15}), and 30 (MV_{Dq30}). Both



(a) Y Quantizer 5

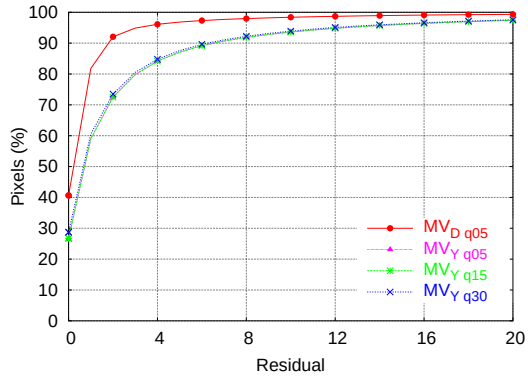


(b) Y Quantizer 15

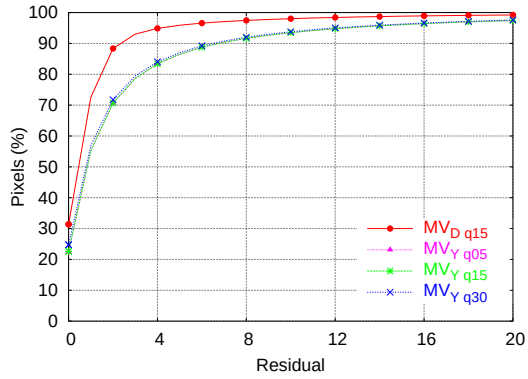


(c) Y Quantizer 30

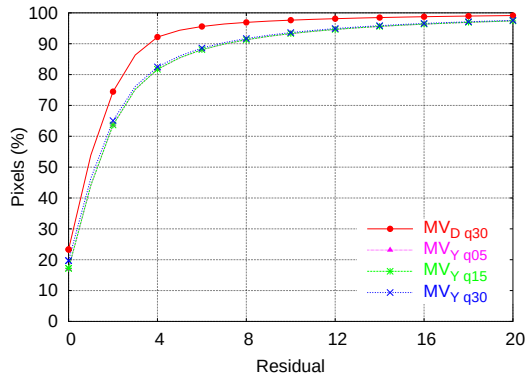
Figure 3.10: Accumulative Residual Histogram of Breakdancers for Luminance



(a) Depth Quantizer 5

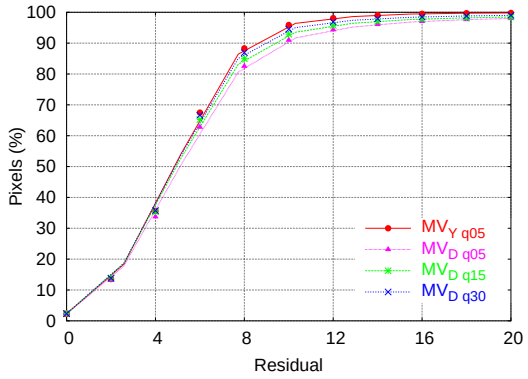


(b) Depth Quantizer 15

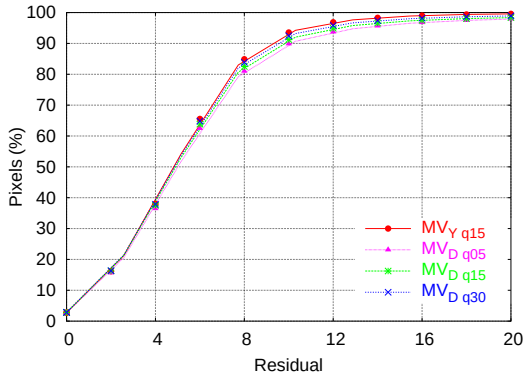


(c) Depth Quantizer 30

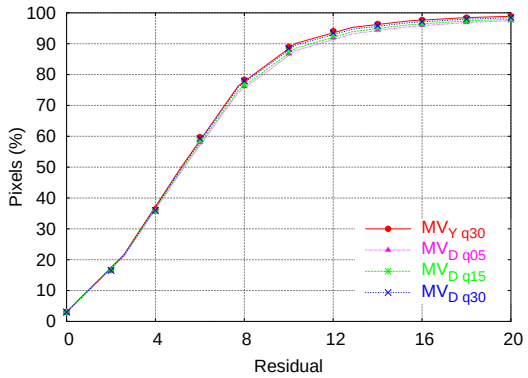
Figure 3.11: Accumulative Residual Histogram of Breakdancers for Depth



(a) Y Quantizer 5

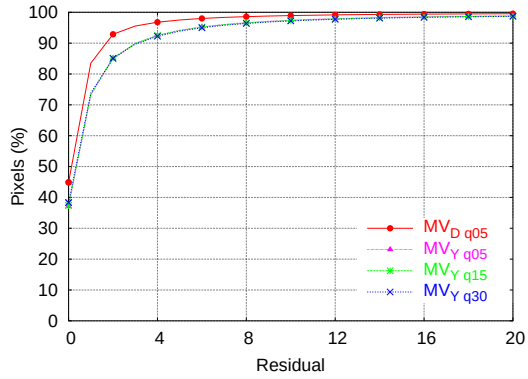


(b) Y Quantizer 15

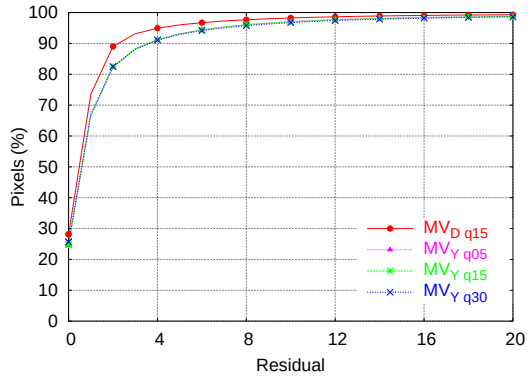


(c) Y Quantizer 30

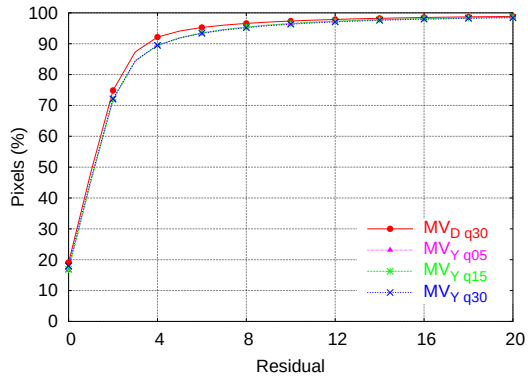
Figure 3.12: Accumulative Residual Histogram of Ballet for Luminance



(a) Depth Quantizer 5

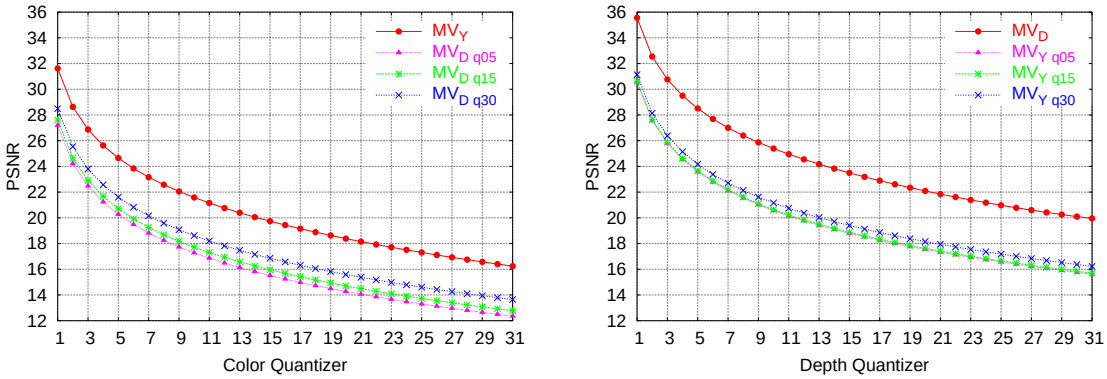


(b) Depth Quantizer 15



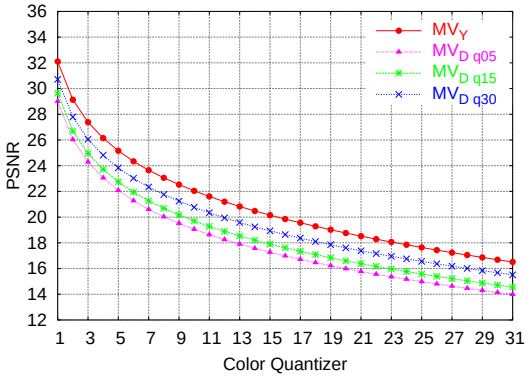
(c) Depth Quantizer 30

Figure 3.13: Accumulative Residual Histogram of Ballet for Depth

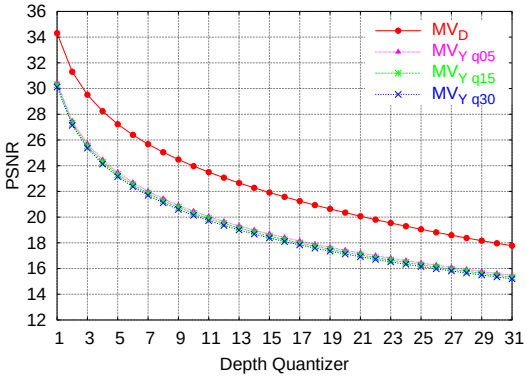


(a) Breakdancers Color

(b) Breakdancers Depth



(c) Ballet Color



(d) Ballet Depth

Figure 3.14: PSNR

Figure 3.12 and 3.13 is averaged over Ballet data of 8 streams of with 99 frames each.

As expected, for luminance prediction using MV_Y yields the best result, and for depth prediction using MV_D is the best. Furthermore, depth has many more pixels with very small residuals than color since depth does not have as much high frequency content to encode as luminance. It is also noticeable that the Ballet data have better prediction overall than the Breakdancers data, which reflects the fact that Breakdancers have faster motion between frames.

Finally, the color quantizers for computing MV_Y has little effect on generating the predicted image for depth (Figures 3.11 and 3.13). This is consistent with Figures 3.5 and 3.7, where MV_Y are similar at various quantizers when compared against MV_D.

PSNR

Figures 3.14a and c show the PSNR of the color predicted images against the original images averaged over 8 streams with 99 frames each for both datasets. For both figures, the X-axis shows the color quantizer, and the Y-axis shows the PSNR for the corresponding color quantizer. Each figure compares the PSNR for MV_Y and MV_D at quantizers 5 (MV_{Dq05}), 15 (MV_{Dq15}), and 30 (MV_{Dq30}). As expected, the PSNR decreases as the color quantizer increases. Also when compared to MV_D at different quantizers, the color predicted image generated using the MV_Y is always better which agrees with the previous results of the accumulative residual histogram for color (Figure 3.10 and 3.12).

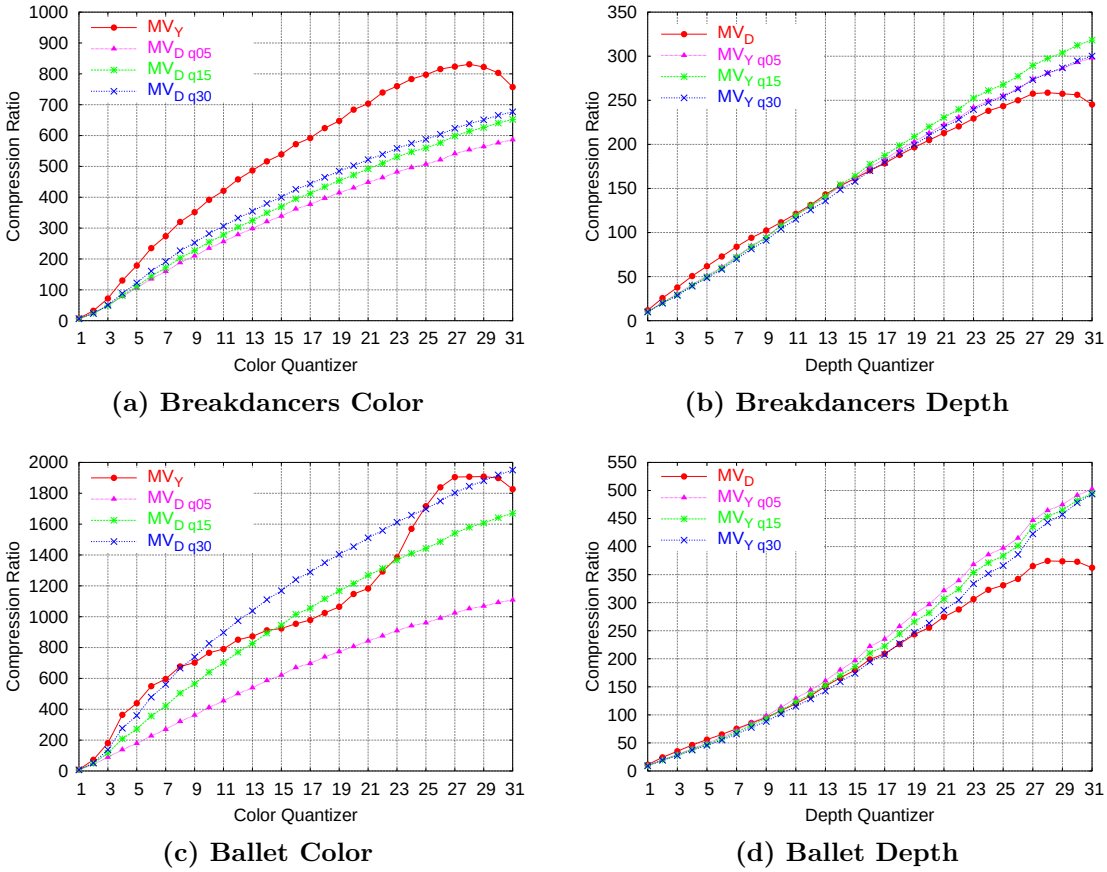
Figures 3.14b and d shows the average PSNR of the depth predicted images for the Breakdancers data (b) and the Ballet data (d) over 8 streams of 99 frames each. For both figures, the X-axis shows the depth quantizer, and the Y-axis shows the PSNR for the corresponding depth quantizer. Each figure compares the PSNR for MV_D and MV_Y at quantizers 5 (MV_{Yq05}), 15 (MV_{Yq15}), and 30 (MV_{Yq30}). As with color, PSNR decrease as the quantizer increase. Also similar to color, the depth predicted image generated using the MV_D is always better than the one generated using MV_Y at different quantizers. This is also consistent with previous accumulative residual histogram for depth (Figure 3.11 and 3.13).

As observed with the accumulative residual histogram, there are little differences between the depth predicted images using MV_Y at various quantizers. However, generating the color predicted images using MV_D is affected by the depth quantizers.

3.2.3 Compression Ratio

Using two motion vectors incurs the extra overhead of specifying an extra motion vector for each block. Therefore, it will only outperform the single motion vector algorithms if the extra motion vector predictions result in residuals that encode more efficiently (i.e. require fewer bits) to make the cost of the additional motion vector worthwhile.

Figures 3.15a and c show the color compression ratio for Breakdancers data and Ballet data averaged over 8 streams of 99 frames each. For both figures, the X-axis shows the color quantizer, and the Y-axis shows the compression ratio for the corresponding color quantizer.



(a) Breakdancers Color **(b) Breakdancers Depth**

(c) Ballet Color **(d) Ballet Depth**

Figure 3.15: Compression Ratio

Each figure compares the compression ratio using the Y motion vector (MV_Y) and the depth motion vector at quantizer 5 (MV_{Dq05}), 15 (MV_{Dq15}), and 30 (MV_{Dq30}). For encoding color with MY_Y, the number of bits used for encoding headers, motion vectors, and color residual is calculated. For encoding color with MV_D, since the motion vector is encoded with depth, only the number of bits used for encoding the color residual and headers is calculated. Even with the extra bits needed to encode the MV_Y, using MV_Y to encode color generally yields a better compression ratio. One exception is for the Ballet data at color quantizers 9 - 25 where increasing the quantizer does not increase the compression ratio very much due to the characteristics of the data. Also, notice that when using MV_D to predict color images, larger depth quantizers lead to better compression. This is because larger depth quantizers for MV_D results in a smaller ΔMV (Figures 3.5 and 3.7), resulting in a predicted image (Figures 3.14) that is closer to the predicted image just using MV_Y.

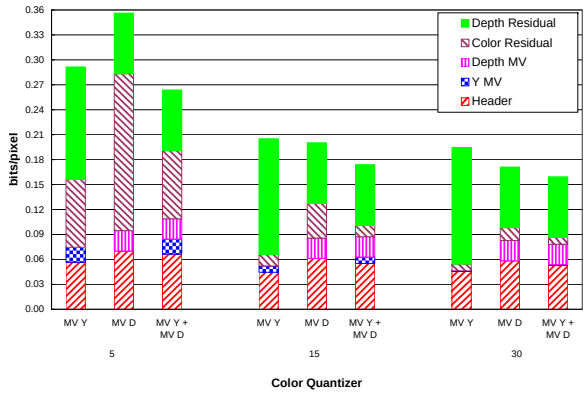
Figures 3.15b and d show the depth compression ratio for Breakdancers data (b) and Ballet

data (d) averaged over 8 streams of 99 frames each. For both figures, the X-axis shows the depth quantizer, and the Y-axis shows the compression ratio for the corresponding depth quantizer. Each figure compares the compression ratio using the depth motion vector (MV_D) and the Y motion vector at quantizer 5 (MV_{Yq05}), 15 (MV_{Yq15}), and 30 (MV_{Yq03}). For encoding depth with MY_D, the number of bits used for encoding headers, motion vectors, and depth residual is calculated. For encoding depth with MV_Y, since the motion vector is encoded with the color, only the number of bits used for encoding the depth residual and headers is calculated. It is interesting that there is no significant difference between using MV_Y and MV_D at small quantizers, but as the quantizers increase using MV_D does worse than MV_Y. However, from Figure 3.11, 3.13, and 3.14, it is clear that using MV_D does a better job of predicting the depth values. This is because, at low quantizers, the bit savings for encoding the residual due to better prediction is larger than the number of bits needed to encode MV_D. However, as the depth quantizers increase, due to the decrease in high frequency information, MV_Y predicts depth better, resulting in more bits needed for MV_D than the bits saved by its prediction.

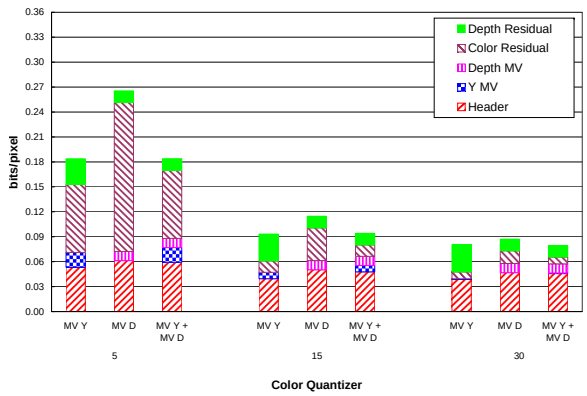
The bit allocations for encoding the header, Y motion vector (MV_Y), depth motion vector (MV_D), color residuals, and depth residuals are shown in Figure 3.16 for Breakdancers data and Figure 3.18 for Ballet data. For each figure, the X-axis indicates the color quantizers for each group – 5, 15, and 30 – and each group has 3 bar graphs; one for encoding with only MV_Y, one for encoding with only MV_D, and one for encoding using both MV_Y and MV_D. The Y-axis indicates the number of bits used to represent a single pixel for the given data. The results at depth quantizer 5, 15, and 30 are shown, where the results were averaged over 8 stream of 99 frames each. Using both MV_Y and MV_D result in the best bitrate for the residuals but an extra motion vector needs to be encoded. Also the increase in bits needed to encode depth residuals using MV_Y compared to using MV_D is larger as the quantizer increases. This also is true for color.

Figure 3.17 (Breakdancers data) and Figure 3.19 (Ballet data) compare the compression ratio of color and depth encoding using MV_Y, MV_D, and MV_Y for color and MV_D for depth at depth quantizer 5, 15, and 30. For each figure, the X-axis shows the color quantizer and the Y-axis shows the compression ratio for the corresponding color quantizer.

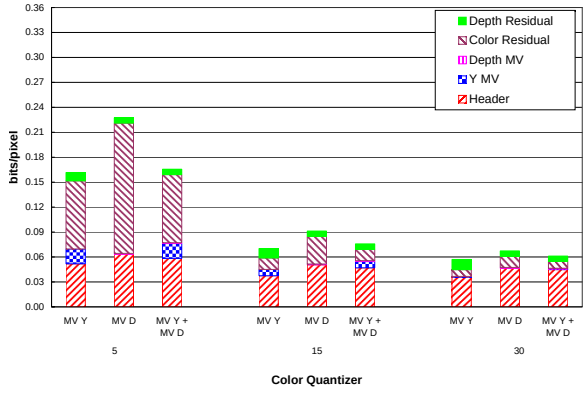
At small depth quantizers, using MV_D and MV_Y yields the best compression. However,



(a) Depth Quantizer 5

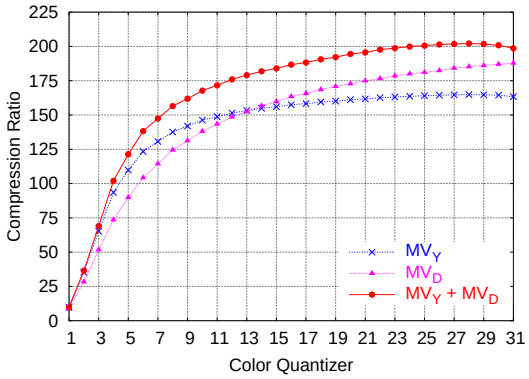


(b) Depth Quantizer 15

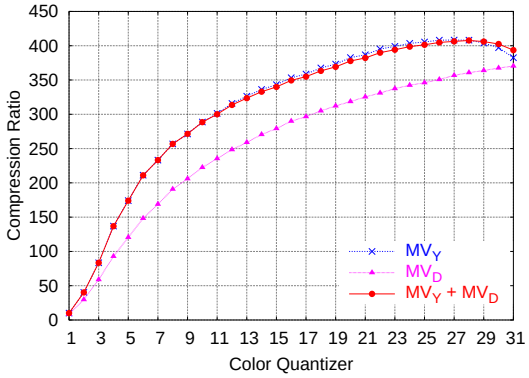


(c) Depth Quantizer 30

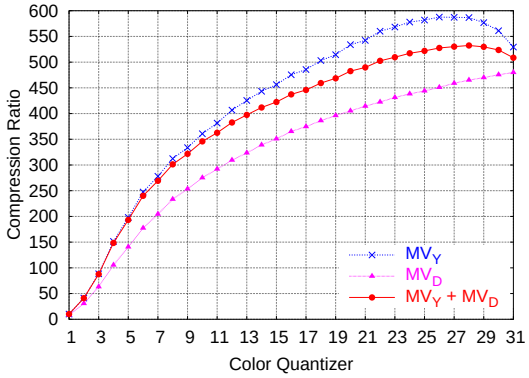
Figure 3.16: Bitrate of Breakdancers



(a) Depth Quantizer 5

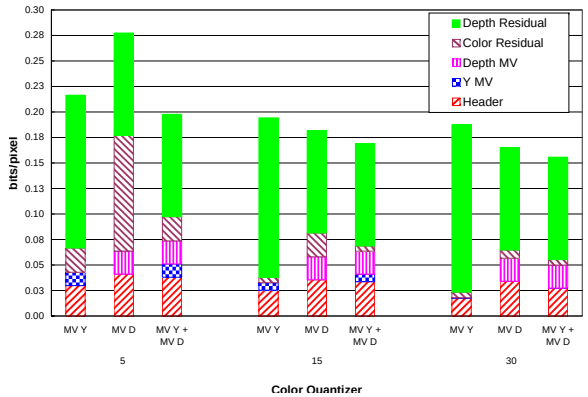


(b) Depth Quantizer 15

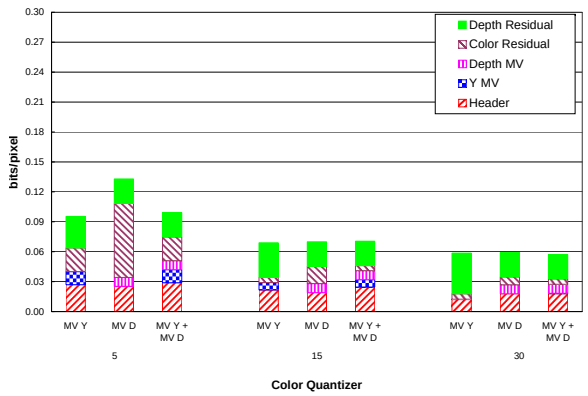


(c) Depth Quantizer 30

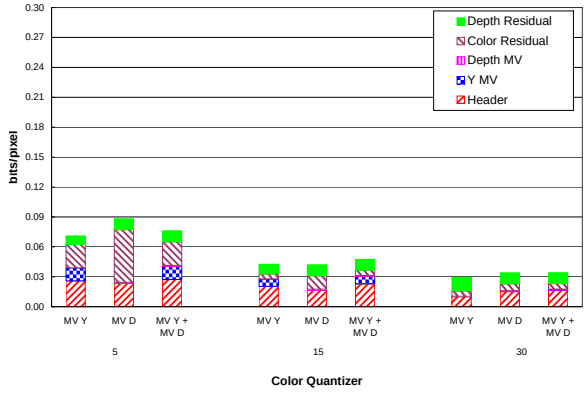
Figure 3.17: Compression Ratio of Breakdancers



(a) Depth Quantizer 5

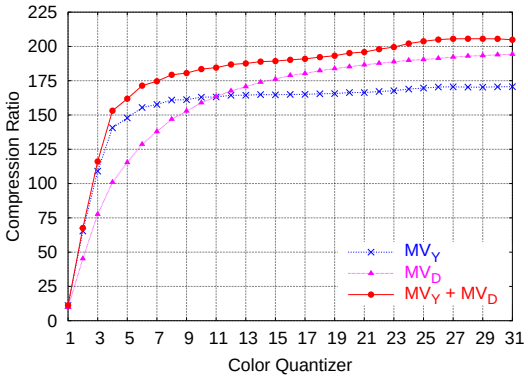


(b) Depth Quantizer 15

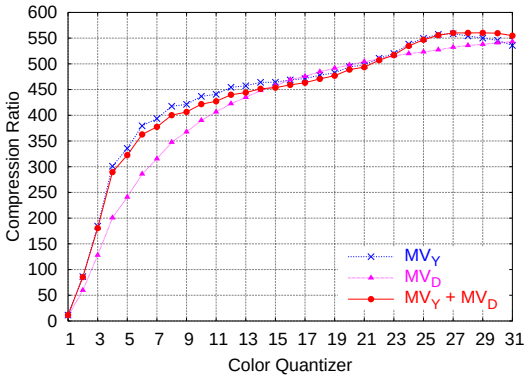


(c) Depth Quantizer 30

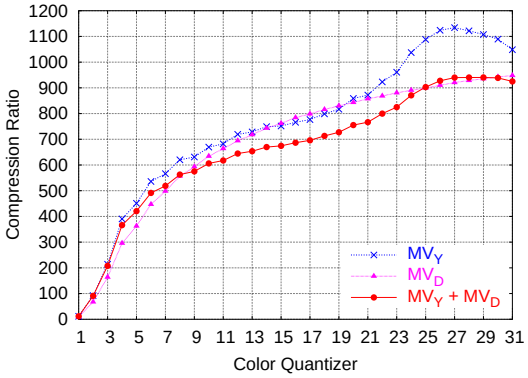
Figure 3.18: Bitrate of Ballet



(a) Depth Quantizer 5



(b) Depth Quantizer 15



(c) Depth Quantizer 30

Figure 3.19: Compression Ratio of Ballet

at high depth quantizers, using MV_Y results in the best compression. Also if only one motion vector can be used, due to limits on computational resources, it is generally better to use MV_Y. But if the required color quality is low and depth quality is high, using MV_D results in a better encoding for two reasons. One is that the high frequency information from color has been sufficiently removed by quantization that MV_D is a decent motion vector to use for color prediction. The other is that more portion of the total bits are used to encode depth, so it is more efficient to improve depth encoding by using MV_D.

3.3 Conclusion

In this chapter, an encoding algorithm for depth streams as intra-streams has been presented. The algorithm extends the motion compensation used for traditional video encoding to also encode depth. Furthermore, different methods of using motion compensation for encoding color and depth were evaluated.

For encoding intra-streams, the experiments show the following:

- For encoding depth at high quality (small quantizer), it is more efficient to use the Y motion vector (MV_Y) to encode color and the depth motion vector (MV_D) to encode depth. The overhead of an extra motion vector per block is smaller than the increase in residual encoding when one motion vector is used.

- For encoding depth with a high quantizer, it is better to use only the color motion vector (MV_Y) because at high depth quantizers, MV_Y is similar enough to MV_D that the depth residuals to be encoded using MV_Y closely match the depth residuals from MV_D. Therefore, encoding an extra motion vector becomes more expensive.

- For encoding with only one motion vector due to limits on computational resources, the motion vector for color (MV_Y) is usually the better choice than the motion vector for depth (MV_D). The exception is for encoding high quality depth and low quality color such that the number of bits required for encoding depth is significantly higher. Accordingly, improving encoding of depth over color results in a more efficient encoding.

Chapter 4

Inter-Stream Encoding

After encoding reference streams as intra-streams, as described in Chapter 3, the next step in compressing multiple depth streams is to encode inter-streams. *Inter-Streams*, also referred to as *P-Streams*, are non-reference streams that exploit the fact that multiple depth streams exhibit strong spatial coherence as they capture the same environment from different angles. Therefore, inter-streams are encoded using the spatial coherence between streams as well as the temporal coherence between frames.

The depth value associated with a pixel in an inter-stream (non-reference stream) can be used to project that pixel into the frame of a reference stream. If the color of the corresponding pixel in the reference stream is similar to the color of the pixel in the inter-stream, encoding of the color information for that pixel in the inter-stream is no longer required. Simply encoding the depth and indicating that the color can be derived from the reference stream is all that is necessary.

This chapter examines how to encode inter-streams using spatial coherence between streams and temporal coherence between frames with in the same stream. Section 4.1 examines how to encode the depth of inter-streams. Then in Section 4.2, the encoding of inter-stream color is presented. The results are presented in Section 4.3 and conclusions in Section 4.4.

4.1 Depth Encoding

Since depth is used for spatial prediction of color in P-Stream encoding, depth values must be encoded independently of color. Therefore, the information that can be used to aid

in encoding of P-Stream depth values are the depth values from the frames of the reference streams (spatial reference frames) and the temporal reference frames with in the same stream. Given these reference frames, there are three different approaches for selecting reference frames to use for encoding: only use temporal reference frames (*Temporal Reference Encoding*), only use spatial reference frames (*Spatial Reference Encoding*), and use both the temporal reference frames and the spatial reference frames (*Spatial and Temporal Reference Encoding*). In this section, these different approaches for encoding depth are explored.

4.1.1 Temporal Reference Encoding

When using only temporal reference frames, P-Stream depth can be encoded much like traditional video encoding using motion compensation. As described in Chapter 2.2, motion compensation utilizes the temporal coherence between frames of the same stream by predicting the current frame from the temporal reference frame using motion vectors. Motion vectors are used to specify a predicted value from the reference frame, and the *residual* – the difference between the predicted value and the actual value to be encoded – is encoded along with the motion vector.

The major difference between depth and color is that depth has only a single channel of information, while traditional video – i.e. color – has three channels. Therefore when encoding depth, only one residual per pixel needs to be encoded instead of three.

Another characteristic of depth values is that high frequency content generally appears on object borders; it is mostly smooth over local regions. Therefore, for depth values of P-Streams, where it does not need to be as accurate as I-Streams since it is not used as a reference stream, subsampling depth values to half resolution before encoding and resampling to full resolution after decoding, just like color chrominance channels, is a viable option.

4.1.2 Spatial Reference Encoding

Another option is to use only spatial reference frames to encode P-Stream depth. The *spatial reference frames* are frames from reference streams at the same point in time, while *temporal reference frames* are frames from the same stream from a different point in time.

Spatial reference frames can be used to encode P-Stream depth by projecting the reference frame pixels to the corresponding pixel in the P-Stream frame to be encoded. Then the corresponding pixel of the P-Stream frame, can use the projected depth value as its predicted value for encoding, and only encode the residual. However, when using this approach, the following possibilities need to be considered:

- No corresponding reference pixels exist for a given pixel.
- Multiple reference pixels with different values correspond to a given pixel.
- Only one reference pixel corresponds to a given pixel but the value does not provide an accurate prediction.

There can be no corresponding reference value for a pixel in the P-Stream frame when the given pixel is not visible from the point of view of the spatial reference frames. This occurs when the pixel is occluded from view in the spatial reference frames or the pixel corresponds to a pixel outside the view of the spatial reference frames. Since these pixels do not have any reference values to predict from, there are two options. One is to encoded the actual depth value, and the other is to generate a predicted value by using the neighboring pixels.

Extra information to distinguish whether the encoded value is the actual depth or the post-prediction residual for a given pixel, needs to be incorporated. Also mixing residual values for some pixles and actual depth values for others within the same block tends to lead to high frequency content in the transform domain, resulting in inefficient encoding.

Using neighboring pixels to generate a predicted value for pixels without any spatial reference values encounters difficulties when the neighboring pixels are part of different objects. Interpolating depth between different objects will generate incorrect depth values, again leading to large residuals and inefficient encoding.

For pixels which have multiple spatial reference pixels with different depth values, the spatial reference depth value itself can be used to select the among possible value predictions. Namely, the smallest reference depth value should be selected as the predicted value. This is because the pixel with the smallest depth value will occlude any other spatial reference pixel with larger depth values that correspond to the same pixel of the frame to be encoded.

For pixels with only one corresponding spatial reference pixel, there is no simple method to accurately detect if the spatial reference pixel is a poor candidate to use as a predicted value. These pixels could result in high residuals which lead to inefficient encoding.

4.1.3 Spatial and Temporal Reference Encoding

As described in the previous section, using only spatial reference frames will most likely result in inefficient encoding. Another approach to improve encoding is to use both spatial and temporal reference frames. Using temporal reference frames additional to spatial reference frames will not help detect poorly predicted values from spatial reference frames, but it will help in cases where no corresponding values exist.

For pixels without predicted depth values from spatial reference frames, the temporal reference frame can be used to provide predicted values. In particular, the depth value of the corresponding pixel in the temporal reference frame can be used as the predicted value to calculate the residual. This will generate a predicted value for each pixel without having to conduct a motion vector search or interpolation of neighboring values. However, a poorly predicted value can be generated when if the object described by the pixel from the temporal reference frame has moved.

4.1.4 Results

In this section, the different results for various P-Stream depth encoding algorithms are presented. The two data sets, Ballet and Breakdancers from [Zitnick et al. 2004], are used for presenting the results of P-Stream encoding throughout this chapter. Each data set has eight streams which are 100 frames long which were captured at 15 fps. The resolution of each frame is 1024×768, and depth is represented in 8 bits. Depth values were computed using the algorithm described in [Zitnick et al. 2004]. Figure 4.1 shows a frame of color and depth from each data set.

Temporal Reference Encoding

Three different encodings of P-Stream depth using only temporal reference frames are compared in this section. They are:

(a) Breakdancers Color

(b) Breakdancers Depth

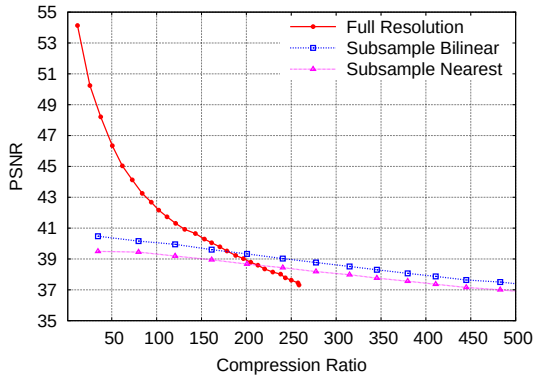
(c) Ballet Color

(d) Ballet Depth

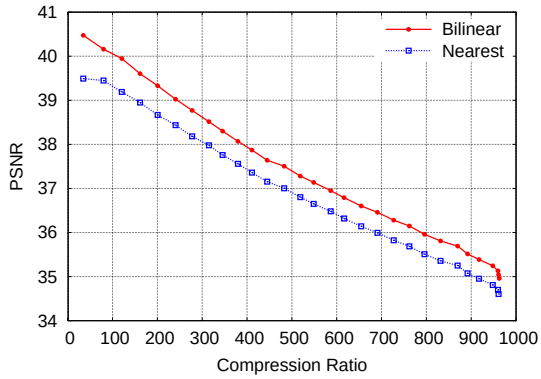
Figure 4.1: Frames from Breakdancers and Ballet Data

- Depth encoded at full resolution of the frame, i.e. 1024×768.
- Depth encoded at half resolution of the frame, i.e. 512×384, by subsampling. To decode depth vales to full resolution, half resolution depth frames were resampled using the nearest neighbor value.
- Depth encoded at half resolution of the frame, i.e. 512×384, by subsampling. To decode depth vales to full resolution, half resolution depth frames were resampled using bilinear interpolation.

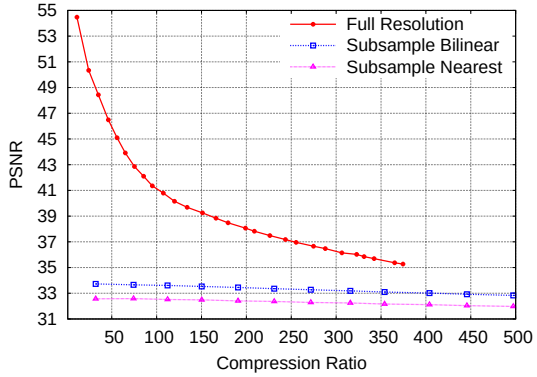
A modified version of the XviD codec was used to encode depth using only temporal reference frames. XviD [XviD] is a well known open source ISO MPEG-4 [MPEG 2004] compliant video codec. XviD was selected mostly because of source code availability, which can be modified without any restrictions. The modified XviD codec is able to encode frames with only one component instead of three.



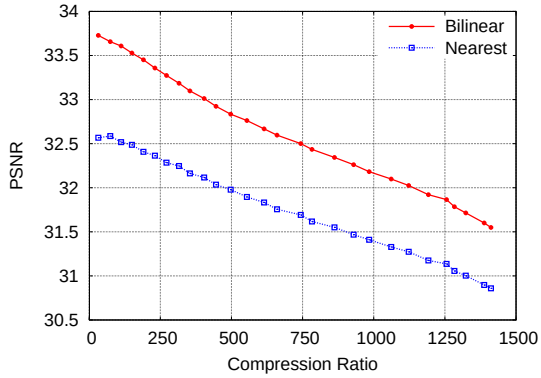
(a) Breakdancers



(b) Breakdancers at Half Resolution



(c) Ballet

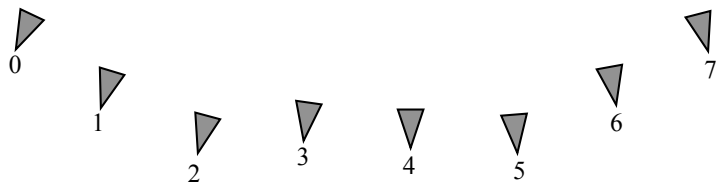


(d) Ballet at Half Resolution

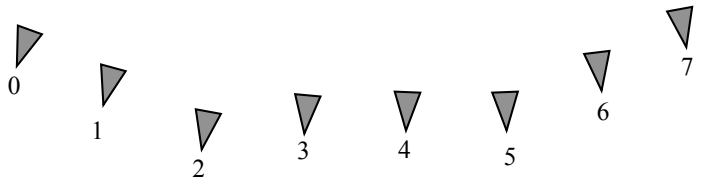
Figure 4.2: P-Stream Depth Temporal Reference Encoding

For all results, all frames were encoded as P-Frames. The previous frame of the encoded frame was selected as the temporal reference frame. All reference frames, spatial and temporal, were encoded as I-Frames. This was done to remove error propagation that happens when a P-Frame is used as a reference frame. Finally, the same quantizer was used for both the I-Frame and the P-Frame.

The results are shown in Figure 4.2. For each figure, the X-axis represents the compression ratio, while the Y-axis represents the PSNR of the decoded depth values. Figure 4.2a compares the results of the Breakdancers data for encoding depth at full resolution (*Full Resolution*), encoding depth at half resolution and then resampled to full resolution using bilinear interpolation (*Subsample Bilinear*), and encoding depth at half resolution and then resampled to full resolution using the nearest neighbor value (*Subsample Nearest*). Figure 4.2b shows the full results for Subsample Bilinear and Subsample Nearest of the Breakdancers data. It shows that encoding at full resolution, instead of encoding at half resolution is better majority of the



(a) Breakdancers



(b) Ballet

Figure 4.3: Stream Positions

time. However, encoding at half resolution enables a much higher compression ratio at the cost of quality (PSNR). Also, as expected, using bilinear interpolation for resampling to full resolution yields better results than using the nearest neighbor.

Figure 4.2c shows the results of the Ballet data for encoding depth at full resolution (*Full Resolution*), encoding depth at half resolution and then resampled to full resolution using bilinear interpolation (*Subsample Bilinear*), and encoding depth at half resolution and then resampled to full resolution using the nearest neighbor value (*Subsample Nearest*). Figure 4.2d shows the full results for Subsample Bilinear and Subsample Nearest of the Ballet data. Similar to the Breakdancers data, encoding at full resolution show better results than encoding at half resolution. A difference in Ballet data from Breakdancers data is that it can be encoded at much higher compression ratio than Breakdancers data. This is primarily due to the fact that Ballet data has slower motion than the Breakdancers data, which leads to better temporal coherence between frames.

Spatial and Temporal Reference Encoding

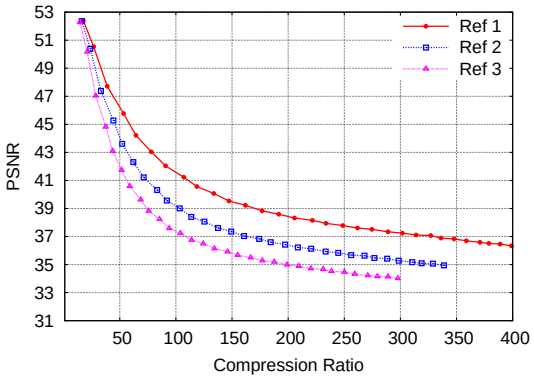
In this section, the results for encoding P-Streams using spatial and temporal reference frames are presented. To utilize spatial reference frames, reference streams need to be specified. Figure 4.3 show the eight stream positions for the Breakdancers and Ballet data set. For all

results in this section, two streams – one on each side of a given stream – were selected as spatial reference streams. For the temporal reference frame, the previous frame in the same stream was selected. Also, all reference frames – spatial and temporal – were encoded using the same modified XviD codec used in the previous section; in addition, all reference frames were encoded as I-Frames with the same quantizer as the encoding frame.

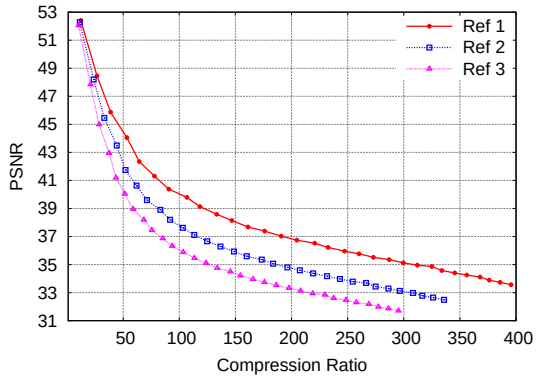
Using the depth value of the spatial reference frames, the spatial reference pixels were projected into the encoding frame. For pixels of the encoding frame with multiple spatial reference depth values, the smaller one was selected as the predicted value. For pixels in the encoding frame with no projected depth value, the depth value of the corresponding pixel from the temporal reference frame was selected as the predicted value. Once all pixels in the encoding frame have been assigned predicted values, the difference between the predicted value and the actual value – i.e. residual – was encoded. Two different methods were evaluated for encoding the residuals. One is based on the discrete wavelet transform (DWT) and the other is based on the discrete cosine transform (DCT). The DWT based codec used was the JPEG2000 [JPEG 2000] encoding, while the DCT based codec used the XviD P-Frame residual encoding.

Figure 4.4 shows the results of each encoding for the Breakdancers and Ballet data sets. In each figure, the X-axis represents the compression ratio, while the Y-axis represents the PSNR of the decoded depth values.

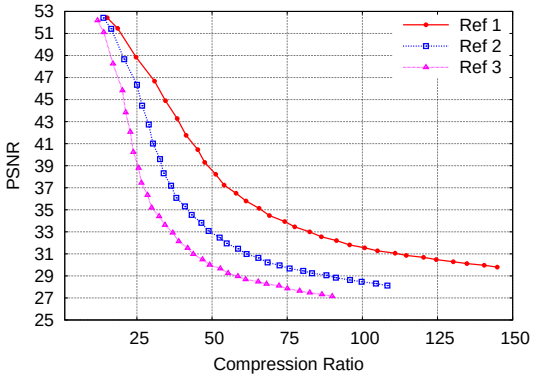
For each encoding, three different configurations of spatial reference streams were compared. The first configuration selected the nearest streams on each side as spatial reference streams (*Ref 1*): From Figure 4.3, stream 1 selected streams 0 and 2, stream 2 selected streams 1 and 3, stream 3 selected streams 2 and 4, stream 4 selected streams 3 and 5, stream 5 selected streams 4 and 6, and stream 6 selected streams 5 and 7 as spatial reference streams. The second configuration selected the second nearest streams on each side as spatial reference streams (*Ref 2*): From Figure 4.3, stream 2 selected streams 0 and 4, stream 3 selected streams 1 and 5, stream 4 selected streams 2 and 6, and stream 5 selected streams 3 and 7 as spatial reference streams. The final configuration selected the third nearest streams on each side as spatial reference streams (*Ref 3*): From Figure 4.3, stream 3 selected streams 0 and 6, and stream 4 selected streams 1 and 7 as spatial reference streams.



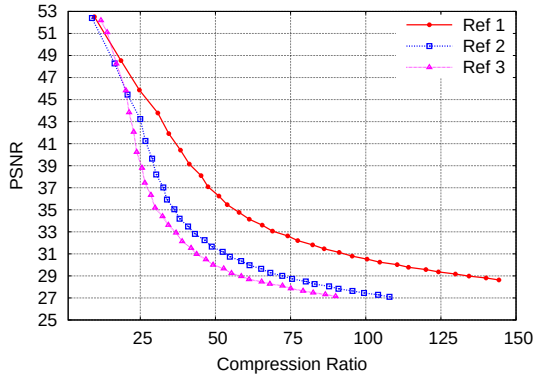
(a) Breakdancers with DWT Encoding



(b) Breakdancers with DCT Encoding



(c) Ballet with DWT Encoding



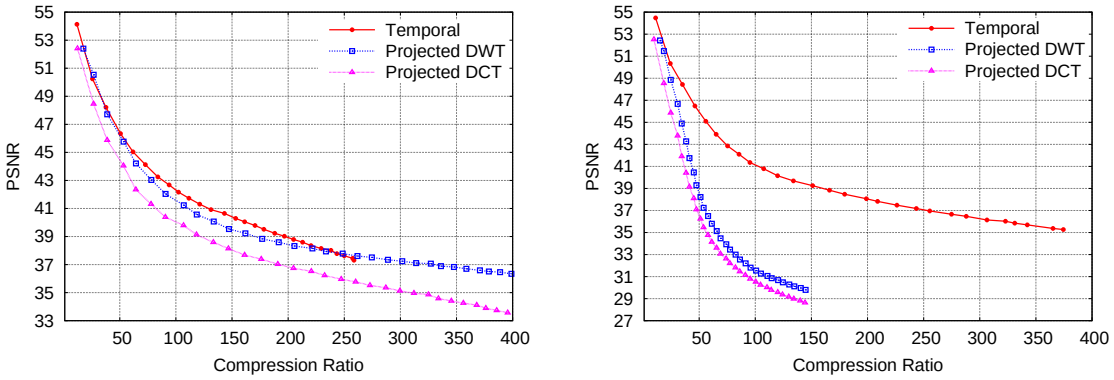
(d) Ballet with DCT Encoding

Figure 4.4: P-Stream Depth Spatial and Temporal Reference Encoding

As seen from Figure 4.4, using the nearest streams as spatial reference streams (Ref 1) always resulted in the best encoding. Also, using the DWT based codec had better results than using the DCT based codec.

4.1.5 Conclusion

The results of various P-Stream depth encodings from Section 4.1.4 are shown in Figure 4.5. For each figure, the X-axis represents the compression ratio, while the Y-axis represents the PSNR of the decoded depth values. The different encodings compared are; depth encoded at full resolution using only temporal reference frames (*Full Resolution* from Figure 4.2) shown as *Temporal*, depth encoded using spatial and temporal reference frames with the residuals encoded using a DWT based codec (*Projected DWT*), and depth encoded using spatial and temporal reference frames with the residuals encoded using a DCT based codec (*Projected DCT*). For both Projected DWT and Projected DCT, the nearest streams were used as spatial



(a) Breakdancers **(b) Ballet**

Figure 4.5: P-Stream Depth Encoding

reference streams (*Ref 1* from Figure 4.4).

For the Breakdancers data set, the Projected DWT encoding is comparable to Temporal encoding (Figure 4.5a), while for the Ballet data set, Temporal encoding results are better (Figure 4.5b). This is primarily due to the fact that Breakdancers data set has faster motion between frames than the Ballet data set. Therefore, the previous frame of the Ballet data set can better predict the current frame than the Breakdancers data set. Also, even though Projected DWT encoding is comparable to Temporal encoding for the Breakdancers data set, this is when the nearest streams are selected as the spatial reference streams. Since all streams need to be encoded, the selected spatial reference streams will not always be the nearest streams for all P-Streams.

Additionally, to use the spatial reference frames to predict depth, the spatial depth values must be accurate. If the depth values are inaccurate, the pixel projection will be incorrect, which will lead to poor corresponding pixels and erroneous reference values.

Because the accuracy of depth values is affected by noise in the original data and lossy encoding of depth, and temporal encoding show better results (Figure 4.5), encoding P-Stream depth using only the temporal reference frames seems to the logical choice. If better compression is needed, P-Stream depth can be encoded using an adaptive approach where depth is encoded in full resolution for high bit rates and half resolution for low bit rates (Figure 4.2).

4.2 Color Encoding

For encoding P-Stream color, the spatial reference frames of the reference streams and the temporal reference frame for the encoding stream is used to predict the color value of the encoding frame. In this section, an algorithm for encoding P-Stream color using multiple reference frames is presented and evaluated. The algorithm does not use any motion compensation for color prediction, and forms predictions on a per pixel basis instead of on a per block basis.

When encoding P-Stream color, the following information is available to better predict the color of a pixel:

- The depth values from the current frame to encode.
- The depth values from the temporal reference frame.
- The depth values from the spatial reference frames.
- The color values from the temporal reference frame.
- The color values from the spatial reference frames.

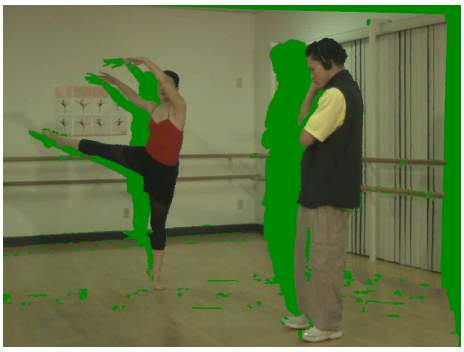
For each pixel color of the encoding frame, the above information is used to predict the pixel color values. Then for each pixel, the *residual* – difference between the predicted color and the actual color – is encoded. Since smaller residuals generally result in more efficient encoding, closer the predicted colors are to the actual color, the more efficient the encoding of the frame.

To predict the color of the current frame to encode at time t ($Color(t)$) from the spatial reference frame of reference stream s, depth of the current frame at time t ($Depth(t)$), depth of the spatial reference frame at time t ($Depth_s(t)$), and the color of the spatial reference frame at time t ($Color_s(t)$) is utilized. For each pixel $(i,\ j)$ in the current frame at time t, the current depth value of the pixel $(i,\ j)$ ($Depth(t, i, j)$) can be used to project the pixel $(i,\ j)$ into reference stream s to find the corresponding pixel $(i_s,\ j_s)$ in $Depth_s(t)$. If the current corresponding depth value in the spatial reference frame ($Depth_s(t, i_s, j_s)$) is close to $Depth(t, i, j)$, the color value of the pixel $(i_s,\ j_s)$ in $Color_s(t)$ ($Color_s(t, i_s, j_s)$) can be used as the predicted color value for pixel $(i,\ j)$. If $Depth(t, i, j)$ is not close to $Depth_s(t, i_s, j_s)$, the pixel $(i,\ j)$ is

(a) Original Frame

(b) Temporal Reference Frame

(c) Stream 2 Spatial Reference Frame



(d) Projected Stream 2 Reference Frame

(e) Stream 5 Spatial Reference Frame

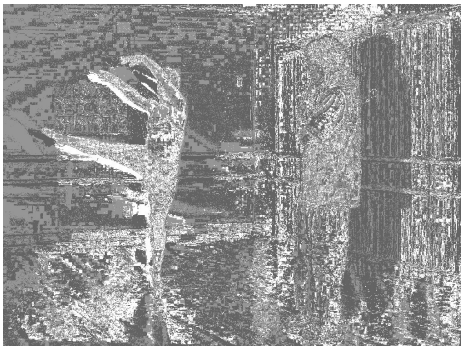
(f) Projected Stream 5 Reference Frame

Figure 4.6: P-Stream Reference Frames

assumed to be occluded in reference stream s, and $Color_s(t, i_s, j_s)$ is not used as a predicted color for pixel $(i,\ j)$ $(Color(t, i, j))$. Also, if pixel $(i,\ j)$ projects outside of reference stream s and no corresponding pixel exists, the $Color(t, i, j)$ does not have a predicted color value from spatial reference stream s.

For predicting the color at pixel $(i,\ j)$ of the current frame to encode at time t $(Color(t, i, j))$ from the temporal reference frame at time $t - 1$, the corresponding pixel $Color(t - 1, i, j)$ is used.

(a) Predicted Image



(b) Reference Mask

(c) Closeup of Predicted Image

(d) Original Frame

Figure 4.7: P-Stream Predicted Image

An example is shown in Figure 4.6. Figure 4.6a shows the encoding frame (frame 10) of stream 3 in the Ballet data set, with stream 2 and 5 selected as the spatial reference streams (Figure 4.3b). Figure 4.6b is the temporal reference frame – the previous frame, frame 9, of stream 3. Figure 4.6c is the tenth frame of stream 2, the spatial reference frame for frame 10 of stream 3. Figure 4.6d is the predicted color values for frame 10 of stream 3 from Figure 4.6c, which was created using the depth values $Depth_3(10)$ to project the pixels into stream 2 to find the corresponding pixels. Pixels marked with green represent pixels that did not have any corresponding pixels, because the depth values were not close enough or the pixels projected outside of stream 2. Figure 4.6e is the tenth frame of stream 5, the other spatial reference frame for frame 10 of stream 3. Figure 4.6f is the predicted color values from Figure 4.6e for frame 10 of stream 3, which was created just like Figure 4.6d.

Figure 4.7a is the resulting predicted image for frame 10 of stream 3 from the spatial reference frames of stream 2 and 5 (Figure 4.6d and f), and the temporal reference frame (Figure 4.6b). A *predicted image* is an image of predicted color values for every pixel, created from

its reference frames to use for encoding. The predicted image (Figure 4.7a) was created by selecting the best predicted color, the predicted color which result in the smallest residual, from the spatial reference frames (Figure 4.6d and f) and the temporal reference frame (Figure 4.6b). Even though the predicted image have sections that does not predict the original frame (Figure 4.7d) well, such as the artificial arm between the two actual arms (Figure 4.7c), it is very similar to the original frame that the residuals calculated using this predicted image is mostly small.

Figure 4.7b is the reference mask of the predicted image (Figure 4.7a) for the original frame (Figure 4.7d). A *reference mask* indicates which reference frame was selected as the predicted color value for the given pixel. The white pixels in the reference mask (Figure 4.7b) indicate that the best predicted color for the corresponding pixels is from spatial reference frame of stream 5 (Figure 4.6f). The light grey pixels imply that the best predicted color values are from spatial reference frame of stream 2 (Figure 4.6d), and the dark grey pixels show that the best predicted values are from the temporal reference frame (Figure 4.6b).

Given a reference mask, a predicted image can be easily created. Therefore, an algorithm for effectively encoding the reference mask can also effectively encode the predicted image. In Section 4.2.1, an algorithm for effectively encoding the reference mask is presented, and the encoding of the residuals are described in Section 4.2.2.

4.2.1 Reference Mask

An uncompressed reference mask, for a frame resolution of 1024×768 which has two spatial reference streams, requires 1536 Kbits ($2 \times 1024 \times 768$) to encode which is not very efficient. However if the reference mask can be encoded efficiently, the predicted image can be encoded efficiently, which result in smaller residuals for effective encoding of the frame. As previously mentioned, the following information is available to be utilized for effective encoding of the reference mask for a P-Stream frame at time t:

- Depth values of the encoding frame at time t ($Depth(t)$).
- Depth values of temporal reference frame at time $t-1$ ($Depth(t-1)$).
- Depth values of spatial reference frames for stream s at time t ($Depth_s(t)$).

- Color of temporal reference frame at time $t-1$ ($Color(t-1)$).
- Color of spatial reference frames for stream s at time t ($Color_s(t-1)$).

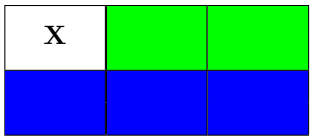
In this section, new primitives used to encode the reference mask using the above information is introduced. Then the reference mask encoding algorithm using these primitives is presented.

Primitives

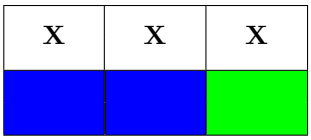
Four primitives are used to encode the reference mask. They are *consensus filter*, *delta depth threshold*, *list of boxes*, and *list of pixels*. Since there are four primitives, only 2 bits are needed to distinguish between the primitives for encoding. Each primitive is described in detail.

Consensus Filter The *consensus filter* selects the consensus of all the spatial reference colors as the predicted color. As shown in Figure 4.6d and f, not all pixels have a corresponding pixel in the spatial reference frames. Therefore, when using the consensus filter, the predicted color for each pixel is selected by the number of existing spatial reference colors available for the given pixel. The consensus filter selects the predicted color for each pixel by the following:

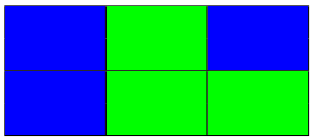
- No spatial reference color exists: Use the temporal reference color as its predicted color for the pixel since it is the only one available.
- Only one spatial reference color exists: If the 1 spatial reference color is similar to the temporal reference color, use the average of the two colors as the predicted color. However, if they are not similar, the predicted color for the pixel cannot be determined using the consensus filter primitive.
- More than one spatial reference color exists: If a consensus can be formed, i.e. majority of the spatial reference colors are similar, the average color of the majority spatial reference colors is used as the predicted color for the pixel. If a consensus cannot be formed, the predicted color for the pixel is unknown.



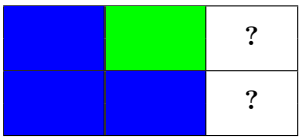
(a) Spatial Reference 1



(b) Spatial Reference 2



(c) Temporal Reference



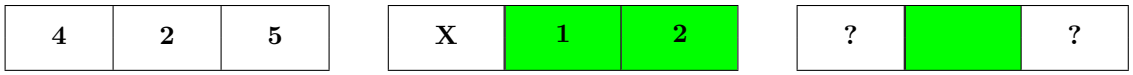
(d) Predicted Image

Figure 4.8: Consensus Filter

Figure 4.8 shows an example of the consensus filter for a frame of 3×2 with two spatial reference streams. Figure 4.8a and b shows the color of the corresponding pixels from the two spatial reference frames. Pixels marked `X` are pixels where no reference color is available for the given pixel. Figure 4.8c shows the temporal reference frame, and Figure 4.8d shows the resulting predicted image using the consensus filter. Pixels marked `?` in the predicted image are pixels where a color cannot be predicted with the consensus filter.

Pixel (0, 0), the pixel at the top-left corner, does not have any spatial reference colors, so the temporal reference color is used as the predicted color for the pixel. Pixel (0, 1), the top-middle pixel, only has one spatial reference color – spatial reference 1 (Figure 4.8a), – which is similar to the temporal reference color. Therefore the predicted color for pixel (0, 1) is the average of the spatial reference color (spatial reference 1) and the temporal reference color. Pixel (0, 2), the top-right pixel, also only has one spatial reference color (Figure 4.8a). But compared to pixel (0, 1) the spatial reference color and the temporal reference color is different. Therefore the predicted color for pixel (0, 2) is unknown (`?`).

All three pixels of the bottom row have more than one spatial reference color. So a consensus needs to be formed for a color to be predicted for a given pixel using the consensus filter. Both spatial reference colors for pixel (1, 0), the bottom-left pixel, are similar and form a consensus. Therefore the average of the two spatial reference colors is used as the predicted color for the pixel (1, 0). Pixel (1, 1), the bottom-middle pixel, also has similar spatial reference colors, but the temporal reference color is different. However, just like pixel (1, 0), the spatial reference colors form a consensus. Therefore the average of the spatial reference



(a) Current Frame Depth **(b) Spatial Reference** **(c) Predicted Image**

Figure 4.9: Delta Depth Threshold

colors is used as the predicted color. The last pixel (1, 2), the bottom-right pixel, does not have a consensus among the spatial reference colors since the two spatial reference colors are different. Therefore the predicted color for the pixel is unknown (`?`), even though the temporal reference color is similar to one of the reference colors.

Encoding the consensus filter only requires 2 bits since it only needs to identify the primitive.

Delta Depth Threshold *Delta depth threshold* specifies a reference frame and a threshold value. For each pixel, the color is predicted from the specified reference frame only if the difference between the depth value of the encoding pixel and the depth value of the corresponding pixel in the reference frame is less than the given threshold.

An example for a frame of 3×1 with the delta depth threshold of `2` is given in Figure 4.9. Figure 4.9a shows the depth values for the current frame to encode ($Depth(t)$). Figure 4.9b shows the depth and color values of the corresponding pixels in the spatial reference frame for the given pixel. The pixel marked with `X` specify that there is no corresponding spatial reference pixel for the given pixel. Figure 4.9c shows the resulting predicted image. Pixels marked with `?` specifies that no predicted color can be determined for the pixel using the delta depth threshold primitive.

The first pixel in the frame (Figure 4.9a) has the depth value of `4`. However, the first pixel of spatial reference frame (Figure 4.9b) indicates that there is no corresponding spatial reference pixel. Therefore the color for the first pixel of the predicted image (Figure 4.9c) is unknown (`?`). For the second pixel, the depth value of the current frame is `2` and the depth value of the spatial reference frame is `1`. Since the difference of these two depth values ($2 - 1 = 1$) is smaller than the given threshold (`2`), the predicted color of the second pixel is same as the color of the spatial reference frame. Finally, the last pixel has the depth value of `5` while the corresponding spatial reference pixel's depth value is `2`. The difference of the depth values ($5 - 2 = 3$) is bigger than

the threshold (2). Therefore the predicted color of the third pixel is also unknown (?).

Delta depth threshold requires $\lceil 8 + \log_2(s+1) \rceil$ bits to encode when there are s number of reference streams – 2 bits to identify the primitive, $\lceil \log_2(s+1) \rceil$ bits to specify the reference frame (1 temporal reference frame and s reference streams), and 6 bits for the threshold. Even though the full range of the threshold requires 8 bits, it can be encoded in 6 bits using differential encoding for each reference. For example, if the previous depth threshold for a given reference was 3 and the current threshold to encode is 10, the actual encoded value would be 7 $(10 - 3)$. Therefore, for encodings where there are two spatial reference streams 10 bits ($\lceil 8 + \log_2(2+1) \rceil = 10$) would be needed to encode the delta depth threshold.

List of Boxes *List of boxes* specifies a reference frame and a list of bounding boxes. The pixels inside the bounding boxes uses the color of the corresponding pixels from the specified reference frame as its predicted color. For spatial reference frames, if there are pixels inside the bounding box that does not have corresponding pixels, the predicted values for these pixels are unknown.

The bits required to encode the list of boxes are:

- 2 bits to identify the primitive.
- $\lceil \log_2(s+1) \rceil$ bits to specify the reference frame when there are s reference streams. So if there are 2 reference streams, 2 bits ($\lceil \log_2(2+1) \rceil$) are needed.
- 4 bits to specify the total number of boxes for a maximum of 17 per primitive; there has to be at least 1 box. If more than 17 boxes need to be specified, multiple list of boxes primitive can be used.
- $2 \times (\lceil \log_2 X \rceil + \lceil \log_2 Y \rceil)$ bits per bounding box when the frame size is $X \times Y$. This is because a bounding box can be specified using the pixel coordinates of the top-left corner and the bottom-right corner. So for a frame size of 1024×768, 40 bits ($2 \times (\lceil \log_2 1024 \rceil + \lceil \log_2 768 \rceil)$) is needed per bounding box.

So for a frame size of 1024×768 which has two spatial reference streams, a list of 3 boxes would require 128 bits ($2 + 2 + 4 + 3 \times 40$).

List of Pixels *List of pixels* specifies a reference frame and a list of pixels. The specified pixels in the list uses the color of the corresponding pixel from the specified reference frame for its predicted color.

The bits required to encode the list of pixels are:

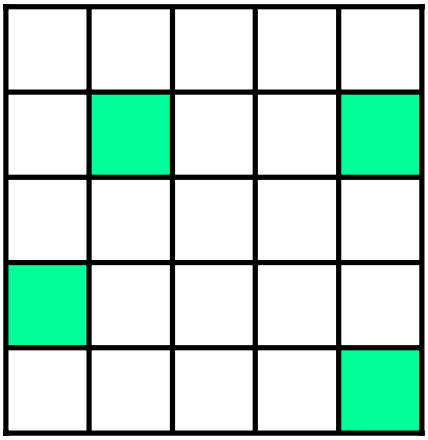
- 2 bits to identify the primitive.
- $\lceil \log_2(s+1) \rceil$ bits to specify the reference frame when there are s reference streams. So if there are 2 reference streams, 2 bits ($\lceil \log_2(2+1) \rceil$) are needed.
- 4 bits to specify the total number of pixels for a maximum of 17 per primitive; there has to be at least 1 pixel. If more than 17 pixels need to be specified, multiple list of pixels primitive can be used.
- $\lceil \log_2 X \rceil + \lceil \log_2 Y \rceil$ bits per pixel when the frame size is $X \times Y$. So for a frame size of 1024×768, 20 bits ($\lceil \log_2 1024 \rceil + \lceil \log_2 768 \rceil$) is needed per pixel.

So for a frame size of 1024×768 which has two spatial reference streams, a list of 5 pixels would require 108 bits ($2 + 2 + 4 + 5 \times 20$).

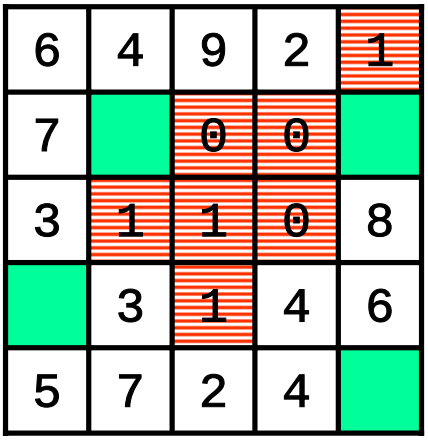
Encoding

The reference mask is encoded as a list of four primitives: consensus filter, delta depth threshold, list of boxes, and list of pixels. The list of primitives are processed in order to assign predicted colors for each pixel. Once a predicted color has been assigned for a pixel, following primitives in the list do not change the predicted color for the pixel: Following primitives only assign predicted colors for pixels that is unknown.

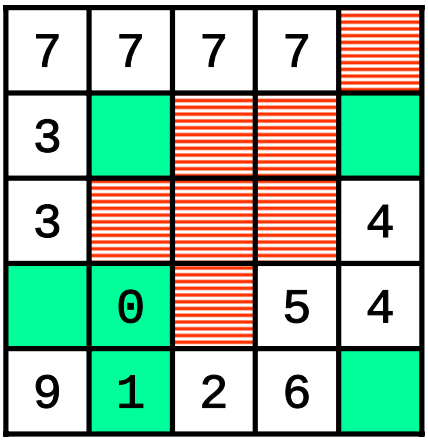
An example for a 5×5 frame with two spatial reference streams is demonstrated in Figure 4.10. Pixel (0, 0) is the upper-left corner of the image, and pixel (4, 4) is the lower-right corner. The number for each pixel, if specified, is the depth difference between the specified reference frame and the encoding frame. For each pixel, if the predicted color is from the temporal reference color it is indicated using solid green, red with horizontal stripes if the first spatial reference color is selected, and blue with vertical stripes if the second spatial reference color is selected.



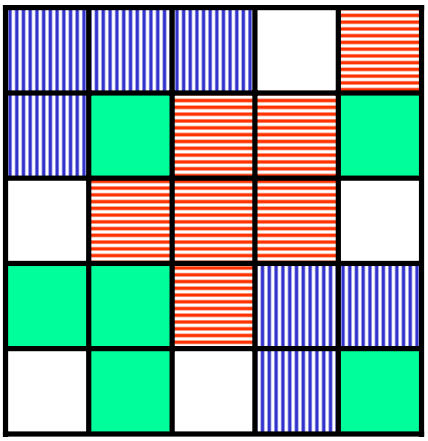
(a) **Temporal, List of Pixels**
(1, 1), (1, 4), (3, 0), (4, 4)



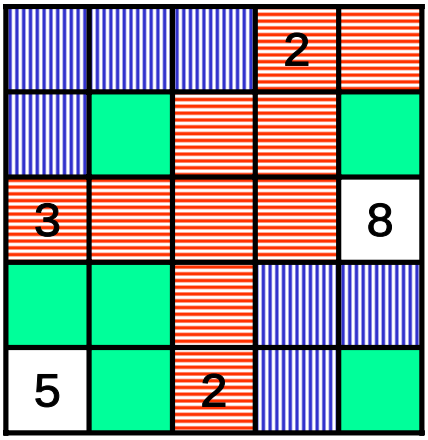
(b) **Spatial 1, Delta Depth Threshold 2**



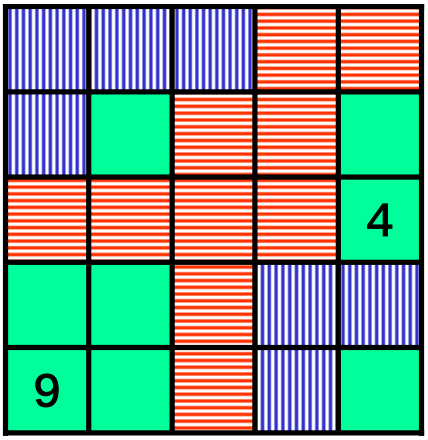
(c) **Temporal, Delta Depth Threshold 2**



(d) **Spatial 2, List of Boxes**
[(0, 0), (2, 1)], [(3, 3), (4, 4)]



(e) **Spatial 1, Delta Depth Threshold 5**



(f) **Temporal, Delta Depth Threshold 255**

Figure 4.10: Reference Mask Encoding

The reference mask for Figure 4.10 is encoded using six primitives. The primitives in order are:

1. List of pixels using temporal reference frame with 4 pixels (1, 1), (1, 4), (3, 0), and (4, 4).
2. Delta depth threshold using spatial reference frame 1 with depth threshold of 2.
3. Delta depth threshold using temporal reference frame with depth threshold of 2.
4. List of boxes using spatial reference frame 2 with 2 boxes [(0, 0), (2, 1)] and [(3, 3), (4, 4)], where a box is expressed as [($top\text{-}left_x$, $top\text{-}left_y$), ($bottom\text{-}right_x$, $bottom\text{-}right_y$)].
5. Delta depth threshold using spatial reference frame 1 with depth threshold of 5.
6. Delta depth threshold using temporal reference frame with depth threshold of 255.

The result of processing the first primitive – list of pixels using temporal reference frame at (1, 1), (1, 4), (3, 0), and (4, 4) – is shown in Figure 4.10a. The four pixels use the color from the corresponding pixels in the temporal reference frame as its predicted color, and all other pixel's predicted color is unknown.

Processing the second primitive, delta depth threshold using spatial reference frame 1 with threshold of 2, results in Figure 4.10b. The numbers for each pixel is the depth difference of the pixel from the corresponding pixel in the spatial reference frame 1. Pixels without the numbers – (1, 1), (1, 4), (3, 0), and (4, 4) – already have predicted colors selected from the first primitive, so there is no need to apply the second primitive for these pixels. Of the unknown pixels, pixels with depth difference of less than 2 – (0, 4), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), and (3, 2) – use the color of the corresponding pixel from spatial reference frame 1 as its predicted color.

Processing the third primitive, delta depth threshold using the temporal reference frame with threshold of 2, results in Figure 4.10c. As with Figure 4.10b, the numbers for each pixel are the depth difference of the pixel from the corresponding pixel in the temporal reference frame. Pixels with depth difference of less than 2 – (3, 1) and (4, 1) – selects the color of the corresponding pixel in the temporal reference frame as its predicted color.

Processing the fourth primitive – list of boxes using spatial reference frame 2 with boxes [(0, 0), (2, 1)] and [(3, 3), (4, 4)] – results in Figure 4.10d. Every pixel in the two bounding boxes that does not have a predicted color – pixels (0, 0), (0, 1), (0, 2), (1, 0), (3, 3), (3, 4), and (4, 3) – uses the color from the corresponding pixel in the spatial reference frame 2 as its predicted color.

Processing the fifth primitive, delta depth threshold using the spatial reference frame 1 with threshold of 5, results in Figure 4.10e. Pixels with depth difference of less than 5 that does not have a predicted color – (0, 3), (2, 0), and (4, 2) – use the color of the corresponding pixel from spatial reference frame 1 as its predicted color.

Processing the final primitive, delta depth threshold using the temporal reference frame with threshold of 255, results in Figure 4.10f. This primitive will assign a predicted color to all pixels that does not have one since the maximum possible depth difference is 254; since depth is represented with 8 bits where the minimum is 1 and maximum is 255. All pixels without predicted colors – (2, 4) and (4, 0) – use the color of the corresponding pixel in the temporal reference frame as its predicted color.

The number of primitives used to encode the reference mask will determine the quality of the encoding. It is even possible to encode the exact reference mask at the cost of encoding efficiency. Also, since the encoding is dependent on the number of primitives and not the resolution of the frame, it is independent of increase in frame resolution.

The reference mask is encoded in two steps: initialization and error correction. The initialization step predicts the color of each pixel by selecting the best primitive possible while minimizing error. The initialization step is done in the following order:

1. Use the consensus filter.

2. Create a list of boxes where temporal reference color should be used as the predicted color.

3. For the remaining pixels, delta depth threshold primitives are used. Delta depth threshold primitives which minimizes error is repeatedly selected until all pixels have a predicted color. Selecting a delta depth threshold of 255 with the temporal reference frame will guarantee all pixels have a predicted color.

The next step, error correction, identifies the pixels with largest error introduced in the initialization step and modify them to reduce error. In this step, pixels with the largest error are identified and either a list of pixels or a list of boxes is created to correct the error. The created primitives are inserted into the list of primitives from the initialization step such that the identified error is corrected.

One way, would be to identify the primitive that introduced the error (*error causing primitive*) for the given pixel, and insert an *error correcting primitive* in the list of primitives before the error causing primitive. This would prevent the error causing primitive from selecting the poor reference color as the predicted color for the pixel, since the predicted color for the pixel would have already been selected by the error correcting primitive.

The error correction can be done until all error pixels are corrected – i.e. lossless encoding of the reference mask. However this will lead to a very ineffective encoding of the reference mask due to the number of primitives needed. An effective encoding should balance the encoding size with the accuracy of the encoding.

Figure 4.11 shows the progression of the predicted image for the original frame (Figure 4.11e), as each primitive is applied. The green pixels are pixels where the predicted colors are unknown after applying the given primitive. Figure 4.11a shows the predicted image after the first primitive, consensus filter, is applied. Figure 4.11b shows the predicted image after the second primitive, list of 16 boxes using temporal reference, is applied. Figure 4.11c shows the predicted image after all the delta depth threshold primitives, total of 23, are applied. Since Figure 4.11c is the predicted image after the initialization step, all pixels should have a predicted color. The last delta depth threshold was the temporal reference frame with the threshold of 255 which will predict a color for any pixel without one. Figure 4.11d is the predicted image after error correction. Three primitives, all list of boxes containing one box, was used in the error correction step. This predicted image would be the final predicted image that would be used for encoding.

Figure 4.11e is the original frame of the P-Stream that needs to be encoded. Figure 4.11f is the predicted image which minimizes the residual for every pixel (*best predicted image*). This image is identical to Figure 4.7a, and can be represented by the reference mask Figure 4.7b.

The best predicted image (Figure 4.11f) can be encoded in 1536 Kbits using a reference

(a) Consensus Filter

(b) List of Temporal Boxes

(c) Delta Depth Threshold

(d) Error Correction

(e) Original Frame

(f) Best Predicted Image

Figure 4.11: Predicted Images

mask (Figure 4.7b). The predicted image in Figure 4.11d does not select the best predicted color – i.e. the predicted color that minimizes the residual – for every pixel, however it only requires 1024 bits to encode: 2 bits for the consensus filter, 648 bits for list of 16 boxes using temporal reference ($2+2+4+16\times40$), 230 bits for 23 delta depth thresholds ($23\times(2+2+6)$), and 144 bits for the 3 list of boxes with a box each, used for error correction ($3\times(2+2+4+40)$). This is an improvement of the encoding efficiency by a factor of 1536, with a tradeoff being the higher residuals that needs to be encoded.

4.2.2 Residual

Once a predicted color for each pixel is selected, the residual – difference between the predicted color and the actual color – for each pixel is encoded. The residuals can be considered correcting the errors introduced by the predicted color. Therefore, if the encoding of the residual is lossless, the decoded image should be exactly the same image as the original image regardless of the predicted image. However, lossless encoding of residuals is very expensive and therefore unpractical. For lossy encoding of residuals, the predicted image effects the encoding efficiency of residuals – better the predicted image, less bits needed to efficiently encode the residuals.

Color is represented in YCbCr color space, one luminance (Y) and two chrominance (Cb, Cr) channels, and the residual for each channel is encoded separately. Similar to other color encoding techniques, the bit allocation for luminance and two chrominance channel is 4 to 1 – for every 4 bits used to encode luminance, each chrominance channel uses 1 bit. Also, the residual is a 9 bit signed integer since it is the difference of each color channel, which is represented as an 8 bit unsigned integer.

Using the encoded reference mask, the residuals are calculated and then encoded using a discrete wavelet transform based codec JPEG2000 [JPEG 2000]. JPEG2000 encoding was selected because of the following:

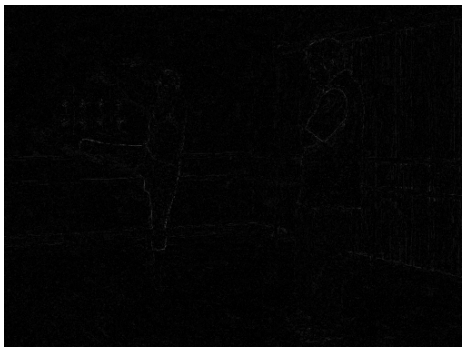
- Support for encoding of 9 bit signed images.
- Quality scalability with target bit rate encoding. JPEG2000 can generate the best distortion rate for a given target encoding length.
- Error resilience. JPEG2000 encoding can be decoded even if it is truncated at arbitrary point in the encoded stream, which is helpful for bit rate control.

Figure 4.12 shows an example of the residual encoding. Figure 4.12c is the original frame to be encoded. Figure 4.12a is the predicted image used to encode Figure 4.12c. Figure 4.12a was created using the algorithm given in Section 4.2.1. Figure 4.12b is the residual for the luminance (Y) channel – difference between the predicted image (Figure 4.12a) and the original frame (Figure 4.12c) – that needs to be encoded. Figure 4.12b does not show the sign of

(a) Predicted Image

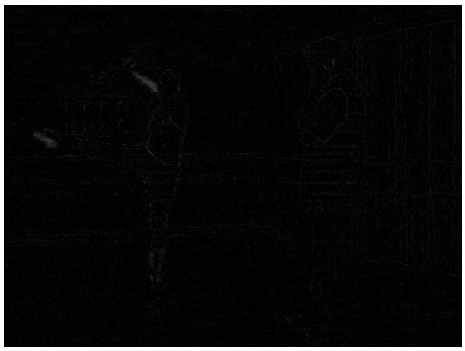
(b) Y Residual (2X)

(c) Original Frame



(d) Cb Residual (4X)

(e) Decoded Frame



(f) Cr Residual (4X)

Figure 4.12: Residual Encoding

the residual but only the magnitude of which is doubled; both residual value of -100 and 100 are shown as intensity value of 200, to better indicate regions where residuals are high. Areas with high residuals are the artificial arm that is shown between the actual arms of the ballerina, artificial right foot, and object boundaries. Figure 4.12d and f is the residuals for the two chrominance channels Cb and Cr. Similar to the luminance residual, only the magnitude is represented but at quadruple its original value – residual of -50 and 50 are shown as 200. Figure 4.12e is the decoded frame of Figure 4.12c from the encoding of the predicted

image (Figure 4.12a) and the residuals (Figure 4.12b, d, and f). It is apparent that the residuals for Cb and Cr channel (Figure 4.12d and f), even at quadruple the original residual, is smaller than the residuals for Y (Figure 4.12b).

4.3 Results

The different results for P-Stream color encoding is presented in this section. Analogous to results presented in this chapter for P-Stream depth encoding (Section 4.1.4), the two data sets, Ballet and Breakdancers from [Zitnick et al. 2004], are used. Each data set has eight streams which are 100 frames long captured at 15 fps. The resolution of each frame is 1024×768, and depth is represented in 8 bits. Depth values were computed using the algorithm described in [Zitnick et al. 2004]. An example frame of color and depth is shown in Figure 4.1.

To encode P-Stream color, spatial reference streams are required. The spatial reference streams used for P-Stream encoding were encoded as I-Streams presented in Chapter 3: The spatial reference streams were encoded using the modified XviD codec presented in Section 3.2 with separate motion vectors for depth and color. All spatial reference frames were encoded as P-Frames with its previous frame as the temporal reference frame which were encoded as I-Frames. This was done to eliminate error propagation when a P-Frame is encoded using another P-Frame as a reference frame.

The previous frame in the encoding stream was selected as the temporal reference frame. The depth of the encoded P-Stream was encoded at full resolution with the modified XviD codec used in Section 4.1.4 since Temporal Reference Encoding showed the best results for encoding P-Stream depth (Figure 4.5). Similar to spatial reference stream encoding, the depth values of the encoding frame were encoded as a P-Frame with the previous frame used as the temporal reference frame which were encoded as an I-Frame. Also the quantizer for spatial reference frame depth, temporal reference frame depth, and the depth of the encoding frame was set to the same value to keep the encoding conditions consistent.

The stream positions of the two data sets, Ballet and Breakdancers, are shown in Figure 4.3. The results for the two middle streams, streams 3 and 4, are presented in this section. They were chosen because different configurations of spatial reference streams can be selected. The

results for reference mask encoding is first shown in Section 4.3.1, followed by results for P-Stream color encoding in Section 4.3.2.

4.3.1 Reference Mask

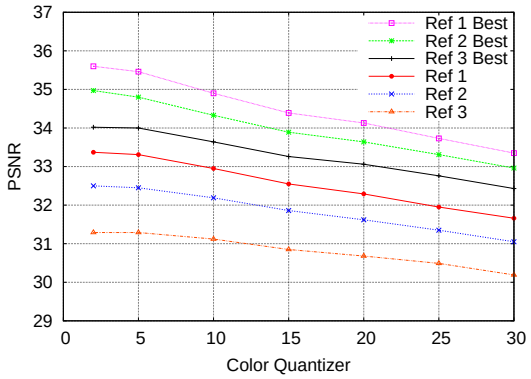
Reference mask encoding results for streams 3 and 4 is presented in this section. For each stream, three different configurations of spatial reference streams were selected. The first used the nearest streams on each side as spatial reference streams (*Ref 1*): From Figure 4.3, stream 3 used streams 2 and 4, and stream 4 used streams 3 and 5 as spatial reference streams. The second used the next nearest streams on each side as spatial reference streams (*Ref 2*): From Figure 4.3, stream 3 used streams 1 and 5, and stream 4 used streams 2 and 6, as spatial reference streams. The final configuration used the third nearest streams on each side as spatial reference streams (*Ref 3*): From Figure 4.3, stream 3 used streams 0 and 6, and stream 4 used streams 1 and 7 as spatial reference streams.

First, the reference mask encoding is compared using the PSNR of the predicted image created from the reference mask. Then the results of reference mask encoding compression ratio are presented.

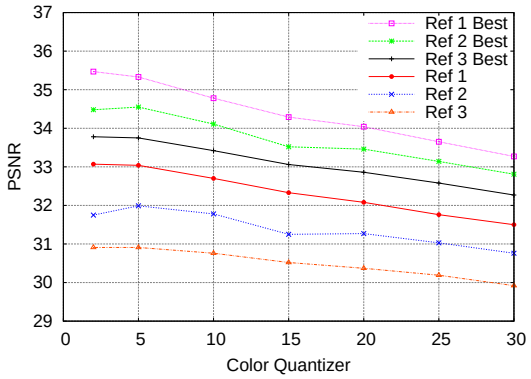
PSNR

The quality of the reference mask encoding can be determined by the quality of the predicted image that is represented by the reference mask. Therefore, the PSNR of the predicted image is compared with the PSNR of the best predicted image to measure how well it predicts the original encoding frame.

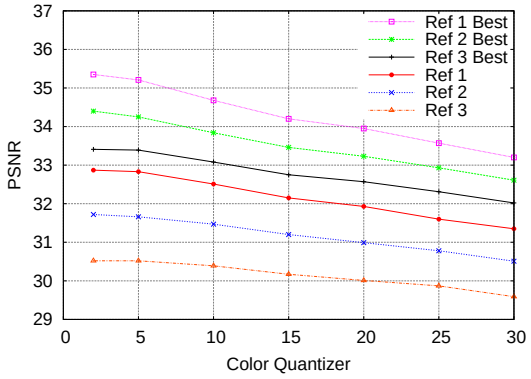
Figure 4.13 compares the PSNR of the predicted images for stream 3 of the Breakdancers data set. For all figures, the X-axis shows the color quantizer for the spatial reference frames, and the Y-axis shows the PSNR of the predicted images. *Ref 1* shows the PSNR of the predicted image using the nearest streams on each side as spatial reference streams (streams 2 and 4), and *Ref 1 Best* shows the PSNR of the best predicted image using the nearest streams on each side as spatial reference streams. *Ref 2* shows the PSNR of the predicted image using the second nearest streams on each side as spatial reference streams (streams 1 and 5), while *Ref 2 Best* shows the PSNR of the best predicted image using the second nearest streams on



(a) Depth Quantizer 5

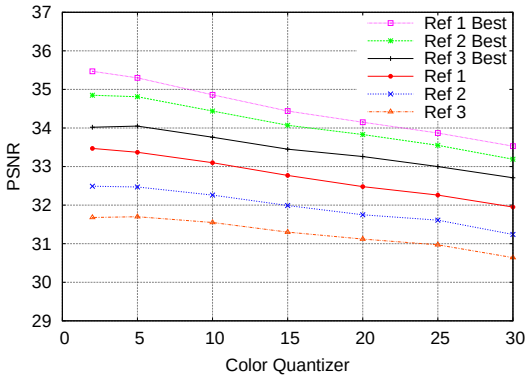


(b) Depth Quantizer 15

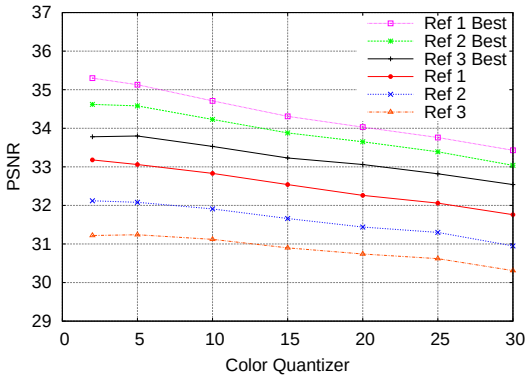


(c) Depth Quantizer 25

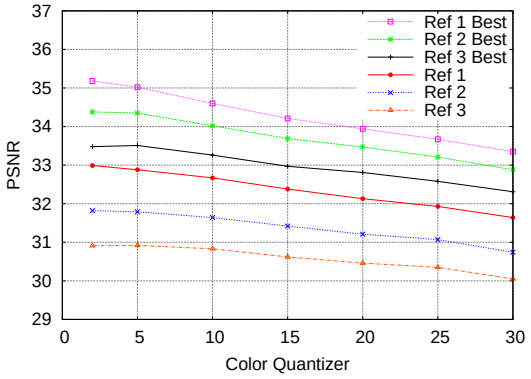
Figure 4.13: Breakdancers Stream 3 Predicted Image PSNR



(a) Depth Quantizer 5



(b) Depth Quantizer 15



(c) Depth Quantizer 25

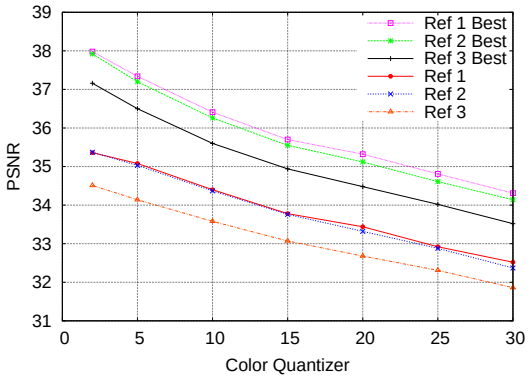
Figure 4.14: Breakdancers Stream 4 Predicted Image PSNR

each side as spatial reference streams. Similarly, *Ref 3* shows the PSNR of the predicted image using the third nearest streams on each side as spatial reference streams (streams 0 and 6), while *Ref 3 Best* shows the PSNR of the best predicted image using the third nearest streams on each side as spatial reference streams. Figure 4.13a shows the results for when the depth quantizer for all frames – spatial reference frames, temporal reference frame, and the encoding frame – were set to 5. Figure 4.13b is the results for when the depth quantizer for all frames were set to 15, and Figure 4.13c is the results for when the depth quantizer for all frames were set to 25.

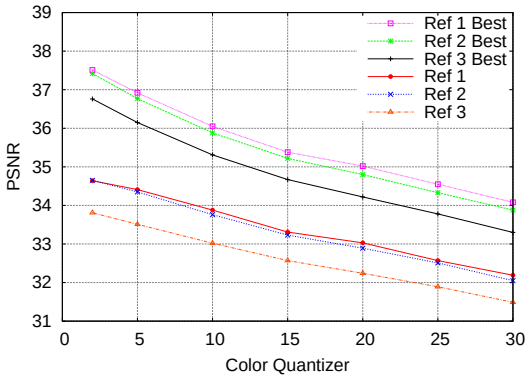
Figure 4.14 compares the PSNR of the predicted images for stream 4 of the Breakdancers data set. For all results in figures Figure 4.14, the X-axis shows the color quantizer for the spatial reference frames, and the Y-axis shows the PSNR of the predicted images. *Ref 1* shows the PSNR of the predicted image using the nearest streams on each side as spatial reference streams (streams 3 and 5), and *Ref 1 Best* shows the PSNR of the best predicted image using the nearest streams on each side as spatial reference streams. *Ref 2* shows the PSNR of the predicted image using the second nearest streams on each side as spatial reference streams (streams 2 and 6), while *Ref 2 Best* shows the PSNR of the best predicted image using the second nearest streams on each side as spatial reference streams. Similarly, *Ref 3* shows the PSNR of the predicted image using the third nearest streams on each side as spatial reference streams (streams 1 and 7), while *Ref 3 Best* shows the PSNR of the best predicted image using the third nearest streams on each side as spatial reference streams. Figure 4.14a shows the results for when the depth quantizer for all frames – spatial reference frames, temporal reference frame, and the encoding frame – were set to 5. Figure 4.14b is the results for when the depth quantizer for all frames were set to 15, and Figure 4.14c is the results for when the depth quantizer for all frames were set to 25.

Results for both stream 3 (Figure 4.13) and stream 4 (Figure 4.14) show that, generally, as the color quantizer and the depth quantizer increase the quality of the predicted image decrease. Also, closer the spatial reference streams, higher the PSNR of predicted images.

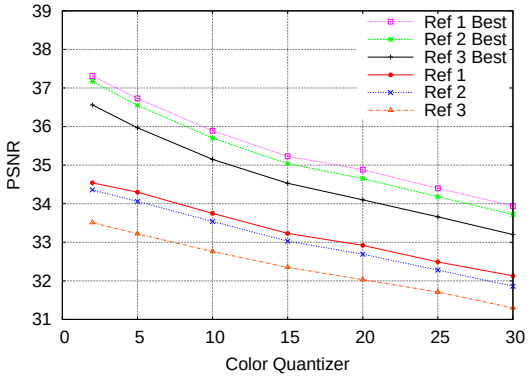
Similar to Figure 4.13 which compares the PSNR of the predicted images for stream 3 of the Breakdancers data set, Figure 4.15 compares the PSNR of the predicted images for stream 3 of the Ballet data set. *Ref 1* shows the results for the PSNR of the predicted image



(a) Depth Quantizer 5

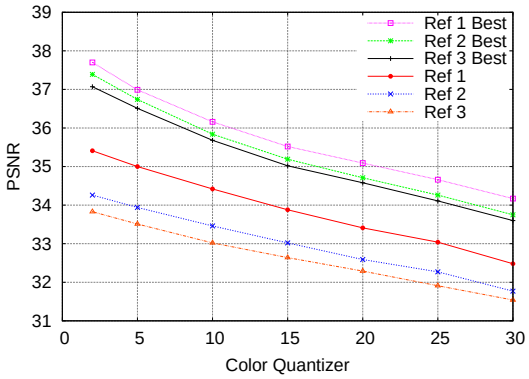


(b) Depth Quantizer 15

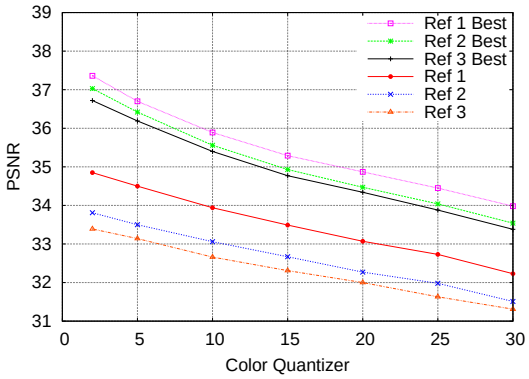


(c) Depth Quantizer 25

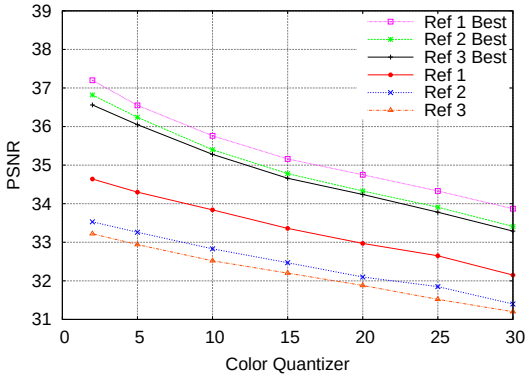
Figure 4.15: Ballet Stream 3 Predicted Image PSNR



(a) Depth Quantizer 5



(b) Depth Quantizer 15



(c) Depth Quantizer 25

Figure 4.16: Ballet Stream 4 Predicted Image PSNR

when the nearest spatial reference streams were selected, *Ref 1 Best* shows the results for the PSNR of the best predicted image when the nearest spatial reference streams were selected, *Ref 2* shows the results for the PSNR of the predicted image when the second nearest spatial reference streams were selected, *Ref 2 Best* shows the results for the PSNR of the best predicted image when the second nearest spatial reference streams were selected, *Ref 3* shows the results for the PSNR of the predicted image when the third nearest spatial reference streams were selected, and *Ref 3 Best* shows the results for the PSNR of the best predicted image when the third nearest spatial reference streams were selected. Also, the X-axis shows the color quantizer for the spatial reference frames, and the Y-axis shows the PSNR of the predicted images. Each figure in Figure 4.15 shows the results for different depth quantizer for all frames – spatial reference frames, temporal reference frame, and the encoding frame: Figure 4.15a is the results for when the depth quantizer for all frames were set to 5, Figure 4.15b is the results for when the depth quantizer for all frames were set to 15, and Figure 4.15c is the results for when the depth quantizer for all frames were set to 25.

Figure 4.16 compares the PSNR of the predicted images for stream 4 of the Ballet data set. For all results, the X-axis shows the color quantizer for the spatial reference frames, and the Y-axis shows the PSNR of the predicted images for all figures. *Ref 1* shows the results for the PSNR of the predicted image when the nearest spatial reference streams were selected, and *Ref 1 Best* shows the results for the PSNR of the best predicted image when the nearest spatial reference streams were selected. *Ref 2* shows the results for the PSNR of the predicted image when the second nearest spatial reference streams were selected, while *Ref 2 Best* shows the results for the PSNR of the best predicted image when the second nearest spatial reference streams were selected. *Ref 3* shows the results for the PSNR of the predicted image when the third nearest spatial reference streams were selected, and *Ref 3 Best* shows the results for the PSNR of the best predicted image when the third nearest spatial reference streams were selected. Figure 4.16a is the results for when the depth quantizer for all frames were set to 5. Figure 4.16b is the results for when the depth quantizer for all frames were set to 15, and Figure 4.16c is the results for when the depth quantizer for all frames were set to 25.

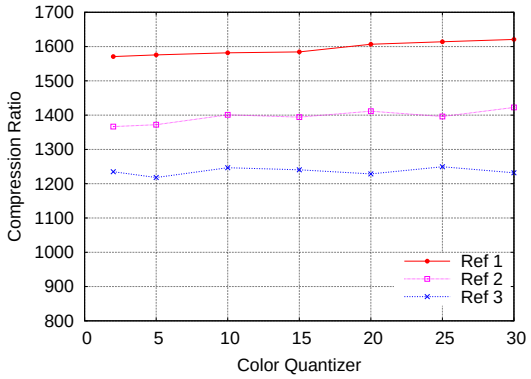
As with the Breakdancers data set, as the color quantizer and the depth quantizer increase the quality of the predicted image decrease. Also, closer the spatial reference streams, higher

the PSNR of the predicted images. However, compared to the Breakdancers data set, for stream 3, the results of *Ref 1* and *Ref 2*, and the results of *Ref 1 Best* and *Ref 2 Best* are very similar at same depth and color quantizers. Also, for stream 4, the results of *Ref 2* and *Ref 3*, and the results of *Ref 1 Best*, *Ref 2 Best*, and *Ref 3 Best* are very similar.

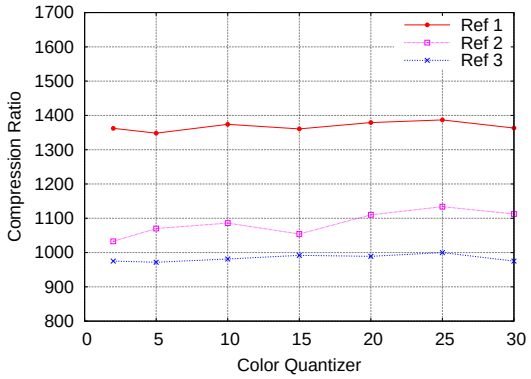
Compression Ratio

In this section, the compression ratio of the reference mask is compared. Figure 4.17 shows the compression ratio of reference mask encoding for stream 3 of the Breakdancers data set. For all figures, the X-axis shows the color quantizer for the spatial reference frames, and the Y-axis shows the compression ratio of the reference masks. *Ref 1* shows the compression ratio for when the nearest streams on each side were selected as spatial reference streams (streams 2 and 4). *Ref 2* shows the compression ratio for using the second nearest streams on each side as spatial reference streams (streams 1 and 5). Similarly, *Ref 3* shows the compression ratio for using the third nearest streams on each side as spatial reference streams (streams 0 and 6). The different figures in Figure 4.17 shows the different results for when the depth quantizer for all frames – spatial reference frames, temporal reference frame, and the encoding frame – differ: Figure 4.17a shows the results for when the depth quantizer were set to 5, Figure 4.17b is the results for when the depth quantizer were set to 15, and Figure 4.17c is the results for when the depth quantizer were set to 25.

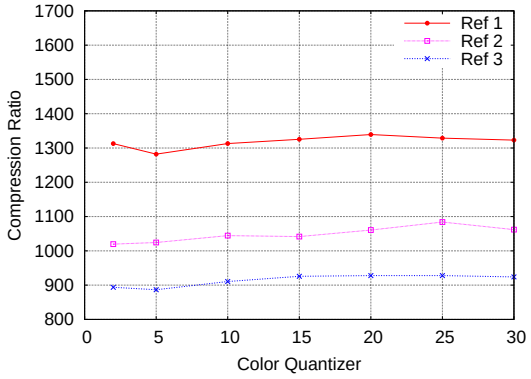
Figure 4.18 shows the compression ratio of reference mask encoding for stream 4 of the Breakdancers data set. For all figures,the X-axis shows the color quantizer for the spatial reference frames, and the Y-axis shows the compression ratio of the reference masks. *Ref 1* shows the compression ratio for when the nearest streams on each side were selected as spatial reference streams (streams 3 and 5). *Ref 2* shows the compression ratio for using the second nearest streams on each side as spatial reference streams (streams 2 and 6). *Ref 3* shows the compression ratio for using the third nearest streams on each side as spatial reference streams (streams 1 and 7). Figure 4.18a shows the results for when the depth quantizer for all frames were set to 5, Figure 4.18b is the results for when the depth quantizer for all frames were set to 15, and Figure 4.18c is the results for when the depth quantizer for all frames were set to 25.



(a) Depth Quantizer 5

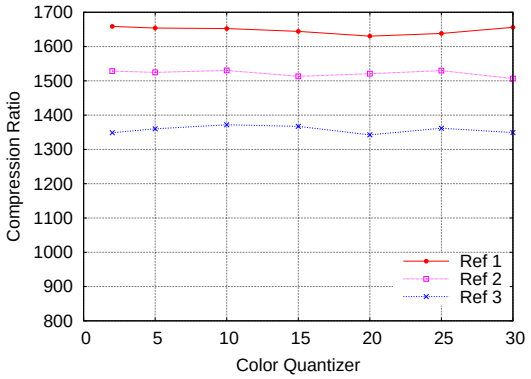


(b) Depth Quantizer 15

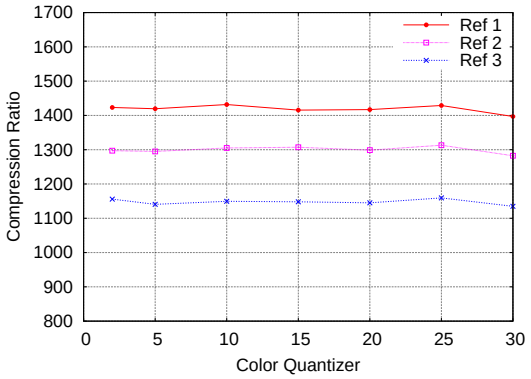


(c) Depth Quantizer 25

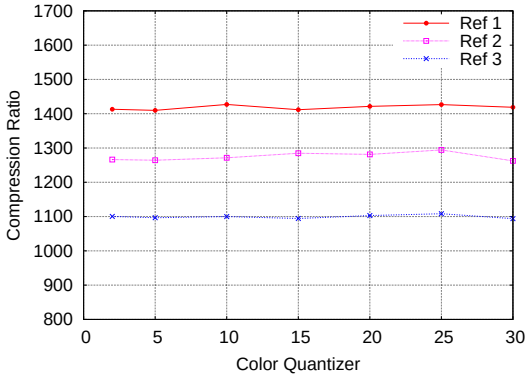
Figure 4.17: Breakdancers Stream 3 Reference Mask Compression Ratio



(a) Depth Quantizer 5



(b) Depth Quantizer 15



(c) Depth Quantizer 25

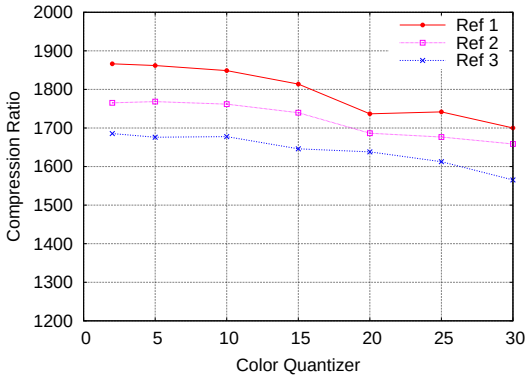
Figure 4.18: Breakdancers Stream 4 Reference Mask Compression Ratio

From both Figure 4.17 and Figure 4.18, as depth quantizers increase or the spatial reference streams are located further, the compression ratio decreases. However, the color quantizers does not seem to influence the compression ratio very much. This indicates that the reference mask is very similar for different quality of the reference color as long as the depth quality is similar. Even though the reference mask is similar, the PSNR of the predicted images differ as the reference color quality changes (Figures 4.13 and 4.14). This is due to the fact that the quality of the reference color used to construct the predicted images differ.

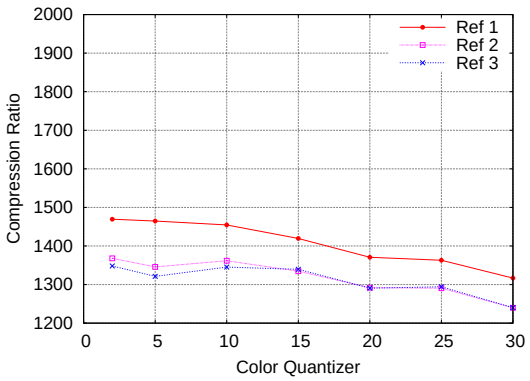
Figure 4.19 shows the compression ratio of reference mask encoding for stream 3 of the Ballet data set. For all figures, the X-axis shows the color quantizer for the spatial reference frames, and the Y-axis shows the compression ratio of the reference masks. *Ref 1* shows the compression ratio for when the nearest streams on each side were selected as spatial reference streams, *Ref 2* shows the compression ratio for using the second nearest streams on each side as spatial reference streams, and *Ref 3* shows the compression ratio for using the third nearest streams on each side as spatial reference streams. The different figures in Figure 4.19 shows the different results for when the depth quantizer for all frames vary: Figure 4.19a is the results for when the depth quantizer were set to 5, Figure 4.19b is the results for when the depth quantizer were set to 15, and Figure 4.19c is the results for when the depth quantizer were set to 25.

Figure 4.20 shows the compression ratio of reference mask encoding for stream 4 of the Ballet data set. For all figures, the X-axis shows the color quantizer for the spatial reference frames, and the Y-axis shows the compression ratio of the reference masks. *Ref 1* shows the compression ratio for when the nearest streams on each side were selected as spatial reference streams, *Ref 2* shows the compression ratio for using the second nearest streams on each side as spatial reference streams, and *Ref 3* shows the compression ratio for using the third nearest streams on each side as spatial reference streams. Figure 4.20a is the results for when the depth quantizer for all frames were set to 5, Figure 4.20b is the results for when the depth quantizer for all frames were set to 15, and Figure 4.20c is the results for when the depth quantizer for all frames were set to 25.

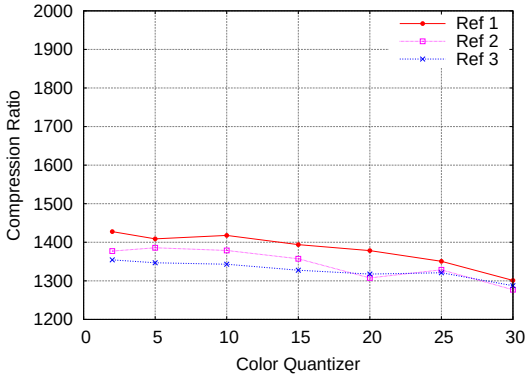
The Ballet data results show higher compression ratios than the Breakdancers data. But similar to the Breakdancers results, the Ballet results demonstrate that the compression ratio



(a) Depth Quantizer 5

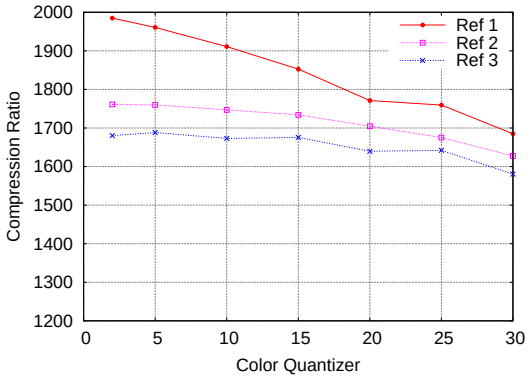


(b) Depth Quantizer 15

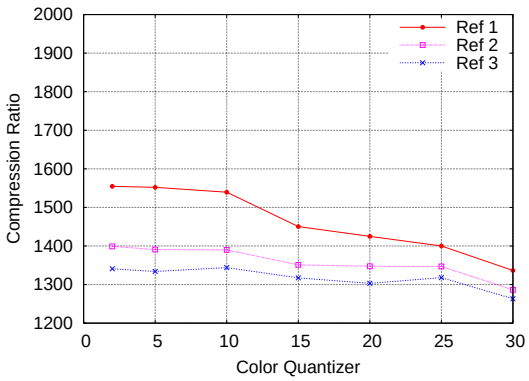


(c) Depth Quantizer 25

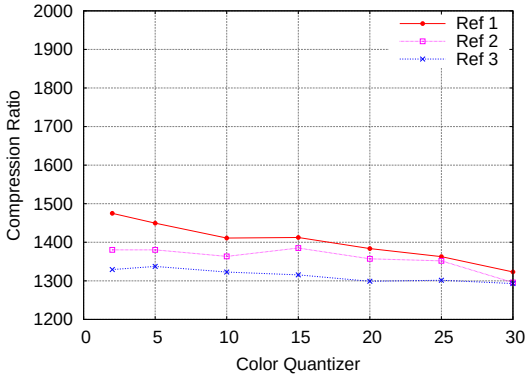
Figure 4.19: Ballet Stream 3 Reference Mask Compression Ratio



(a) Depth Quantizer 5



(b) Depth Quantizer 15



(c) Depth Quantizer 25

Figure 4.20: Ballet Stream 4 Reference Mask Compression Ratio

increases as the depth quantizers decrease or closer spatial reference streams are selected.

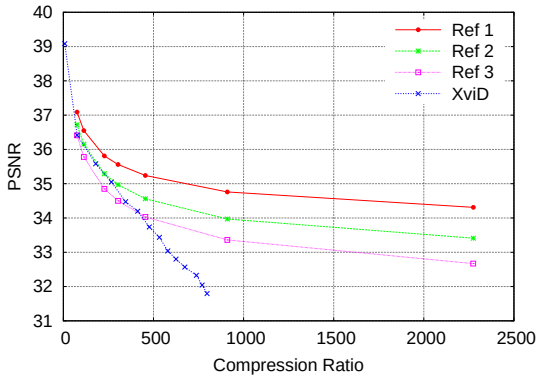
4.3.2 Color Encoding

In this section, the P-Stream color encoding of stream 3 and 4 is compared with XviD encoding. For XviD encoding, to keep it consistent with P-Stream color encoding and to eliminate error propagation when a P-Frame is encoded using a P-Frame as a reference, all frames were encoded as P-Frames with the previous frame always encoded as an I-Frame and used as the temporal reference frame.

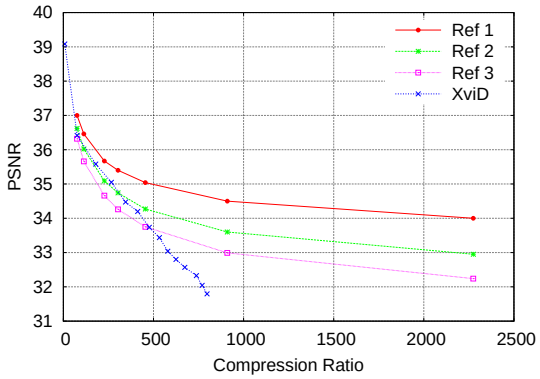
For spatial reference streams, three different configuration were selected. The first used the nearest streams on each side as spatial reference streams (*Ref 1*): From Figure 4.3, stream 3 used streams 2 and 4, and stream 4 used streams 3 and 5 as spatial reference streams. The second used the next nearest streams on each side as spatial reference streams (*Ref 2*): From Figure 4.3, stream 3 used streams 1 and 5, and stream 4 used streams 2 and 6, as spatial reference streams. The final configuration used the third nearest streams on each side as spatial reference streams (*Ref 3*): From Figure 4.3, stream 3 used streams 0 and 6, and stream 4 used streams 1 and 7 as spatial reference streams.

Figure 4.21, Figure 4.23, and Figure 4.25 shows the different encoding results for stream 3 of the Breakdancers data set. Figure 4.21 shows the results for when the color quantizer of the reference color set to 5. Figure 4.23 shows the results for when the color quantizer of the reference color was set to 15, and Figure 4.25 for when it was set to 25. The encoding results for the depth quantizers for all frames – spatial reference frames, temporal reference frame, and the encoding frame – set to 5 (Figure 4.21a, Figure 4.23a, Figure 4.25a), 15 (Figure 4.21b, Figure 4.23b, Figure 4.25b), and 25 (Figure 4.21c, Figure 4.23c, Figure 4.25c) are given in each figure. For all figures, the X-axis shows the compression ratio, and the Y-axis shows the PSNR of the decoded frame. Also, *Ref 1* shows the result for when the nearest streams on each side were selected as spatial reference streams, *Ref 2* shows the result for using the second nearest streams on each side as spatial reference streams, *Ref 3* shows the result for using the third nearest streams on each side as spatial reference streams, and *XviD* shows the result for encoding the stream using the XviD codec.

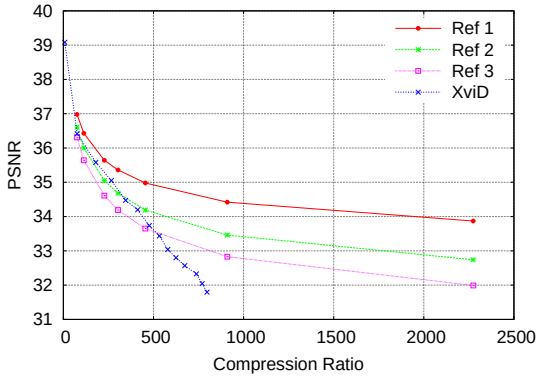
Figure 4.22, Figure 4.24, and Figure 4.26 shows the different encoding results for stream 4



(a) Depth Quantizer 5

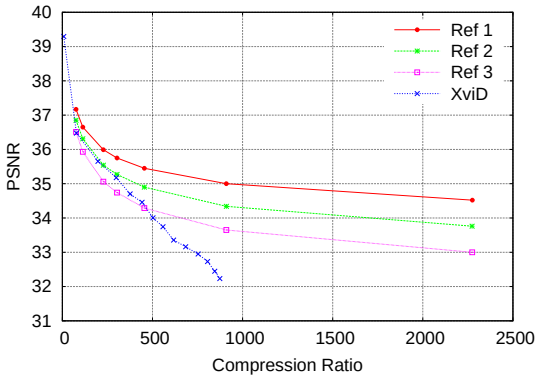


(b) Depth Quantizer 15

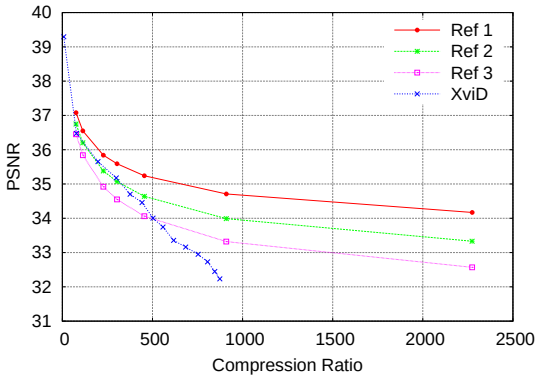


(c) Depth Quantizer 25

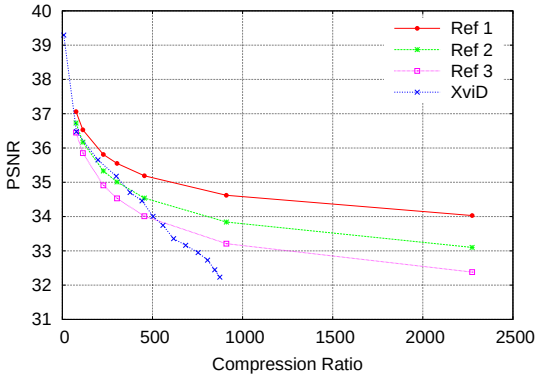
Figure 4.21: Breakdancers Stream 3 Color Quantizer 5



(a) Depth Quantizer 5

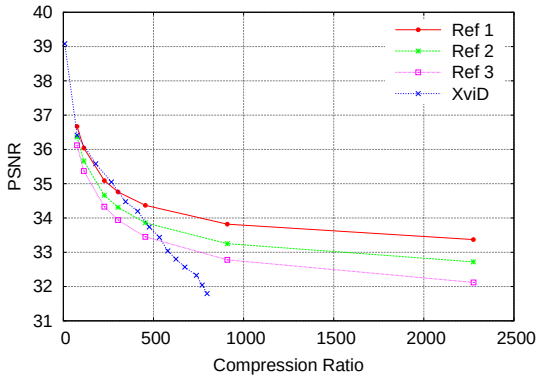


(b) Depth Quantizer 15

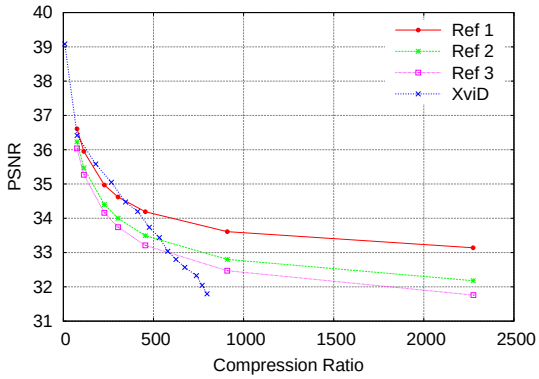


(c) Depth Quantizer 25

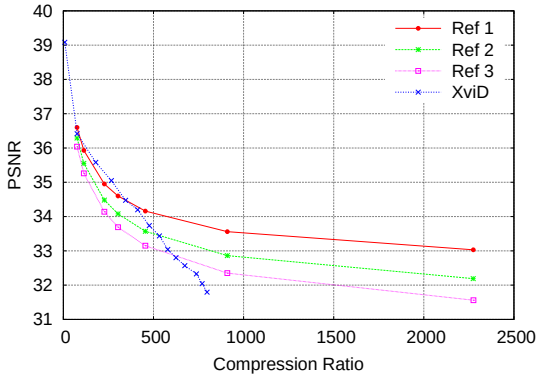
Figure 4.22: Breakdancers Stream 4 Color Quantizer 5



(a) Depth Quantizer 5

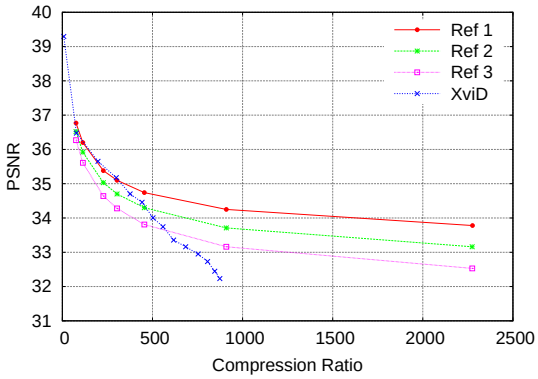


(b) Depth Quantizer 15

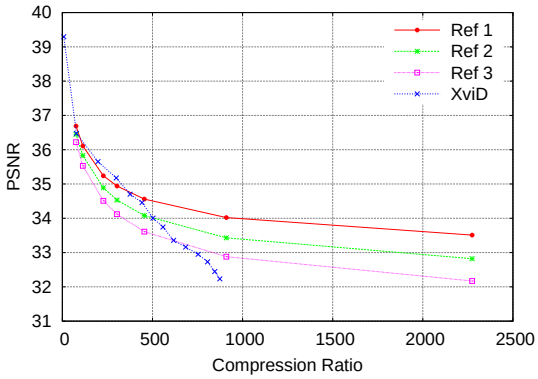


(c) Depth Quantizer 25

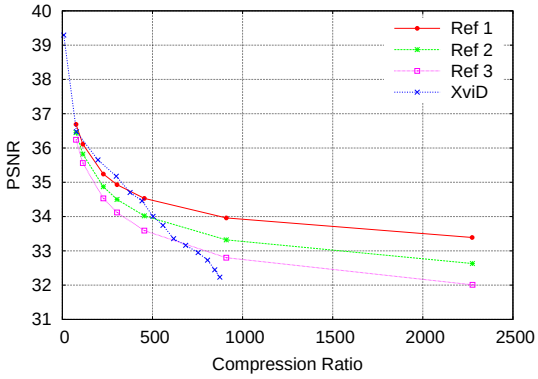
Figure 4.23: Breakdancers Stream 3 Color Quantizer 15



(a) Depth Quantizer 5

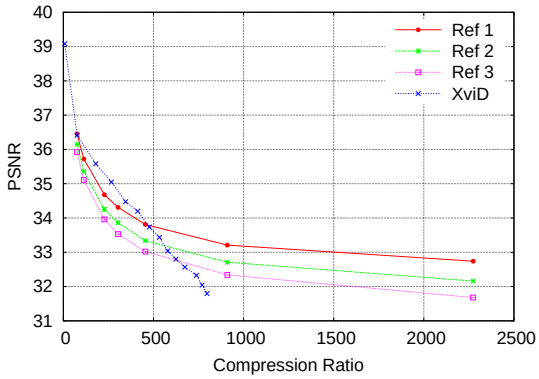


(b) Depth Quantizer 15

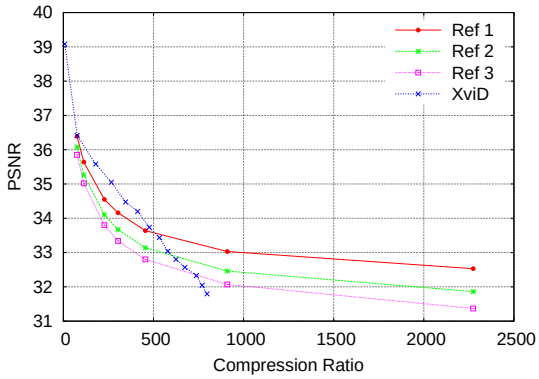


(c) Depth Quantizer 25

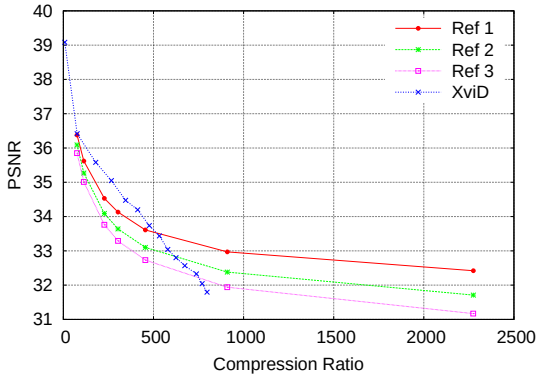
Figure 4.24: Breakdancers Stream 4 Color Quantizer 15



(a) Depth Quantizer 5

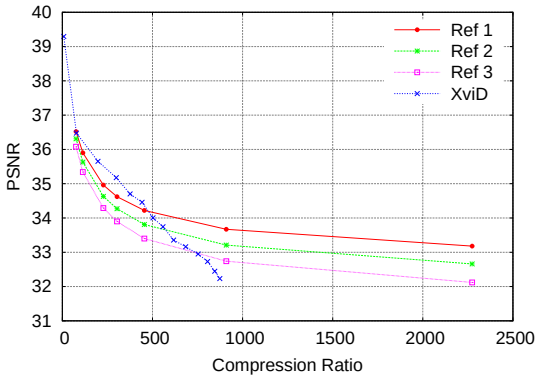


(b) Depth Quantizer 15

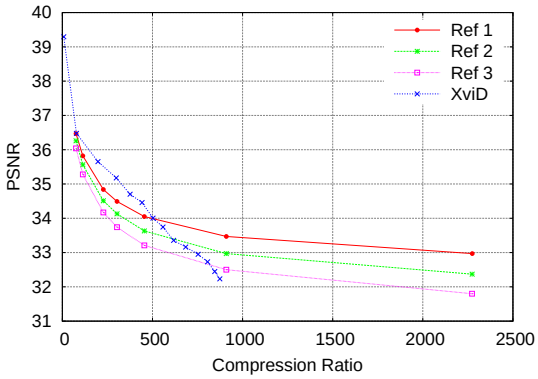


(c) Depth Quantizer 25

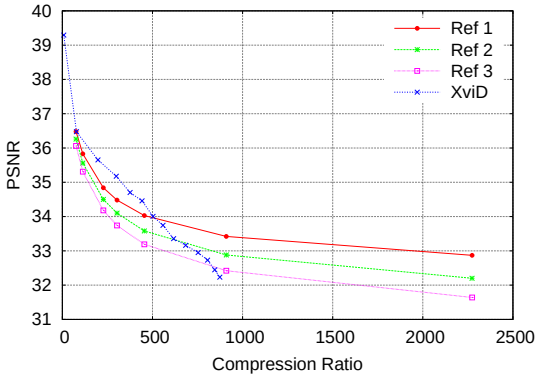
Figure 4.25: Breakdancers Stream 3 Color Quantizer 25



(a) Depth Quantizer 5



(b) Depth Quantizer 15



(c) Depth Quantizer 25

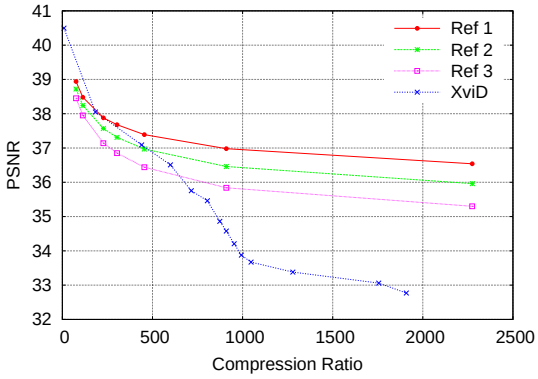
Figure 4.26: Breakdancers Stream 4 Color Quantizer 25

of the Breakdancers data set. Figure 4.22 shows the results for when the color quantizer of the reference color was set to 5. Figure 4.24 shows the results for when the color quantizer of the reference color was set to 15. Figure 4.26 for when the color quantizer of the reference color was set to 25. The encoding results for the depth quantizers for all frames set to 5 (Figure 4.22a, Figure 4.24a, Figure 4.26a), 15 (Figure 4.22b, Figure 4.24b, Figure 4.26b), and 25 (Figure 4.22c, Figure 4.24c, Figure 4.26c) are given in each figure. For all figures, the X-axis shows the compression ratio, and the Y-axis shows the PSNR of the decoded frame. Also, *Ref 1* shows the result for when the nearest streams on each side were selected as spatial reference streams, *Ref 2* shows the result for using the second nearest streams on each side as spatial reference streams, *Ref 3* shows the result for using the third nearest streams on each side as spatial reference streams, and *XviD* shows the result for encoding the stream using the XviD codec.

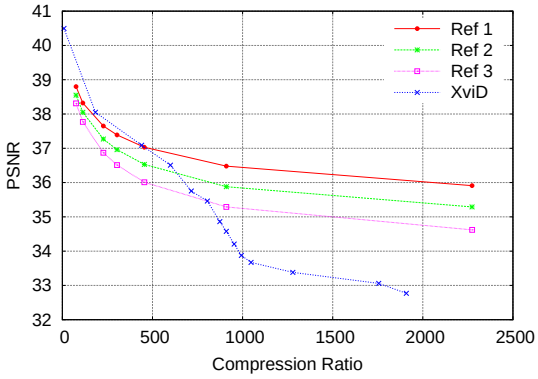
All *XviD* results in Figure 4.21, Figure 4.23, and Figure 4.25 are the same since the XviD encoding is independent of reference color quality, depth quality, or spatial reference streams. This applies to the *XviD* results in Figure 4.22, Figure 4.24, and Figure 4.26 as well.

At low compression ratios (below 500), the P-Stream encoding is slighter better than the XviD encoding when using high quality reference colors (color quantizer 5) for different depth qualities (Figure 4.21, Figure 4.22). For medium quality reference colors (color quantizer 15), only the *Ref 1* is comparable to *XviD*; other P-Stream encodings have lower PSNR (Figure 4.23, Figure 4.24b). At high color quantizer of 25 for reference colors, XviD encoding has higher PSNR than all P-Stream encodings. However at high compression ratios, the P-Stream encoding achieves much higher compression ratios for the same PSNR of the XviD encoding.

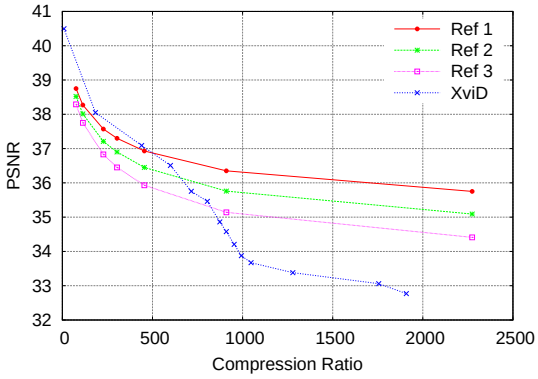
Figure 4.27, Figure 4.28, Figure 4.29, Figure 4.30, Figure 4.31, and Figure 4.32 shows the different encoding results for the Ballet data set: Figure 4.27, Figure 4.29, and Figure 4.31 for stream 3, and Figure 4.28, Figure 4.30, and Figure 4.32 for stream 4. Figure 4.27 and Figure 4.28 shows the results for when the color quantizer of the reference color was set to 5. Figure 4.29 and Figure 4.30 shows the results for when the color quantizer of the reference color was set to 15, and Figure 4.31 and Figure 4.32 for when it was set to 25. Figure 4.27a, Figure 4.28a, Figure 4.29a, Figure 4.30a, Figure 4.31a, and Figure 4.32a show the encoding results for different depth quantizers for all frames – spatial reference frames, temporal reference frame, and the encoding frame – set to 5. Figure 4.27b, Figure 4.28b,



(a) Depth Quantizer 5

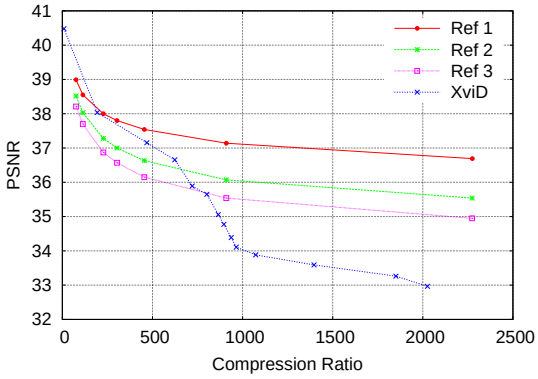


(b) Depth Quantizer 15

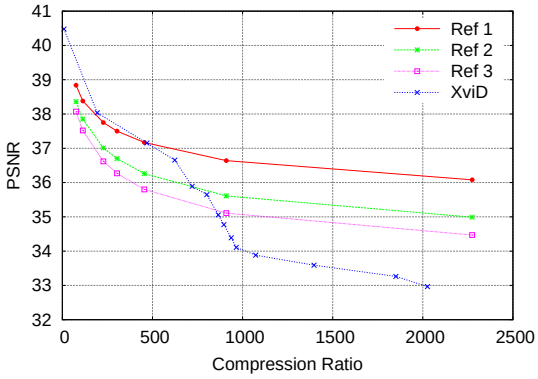


(c) Depth Quantizer 25

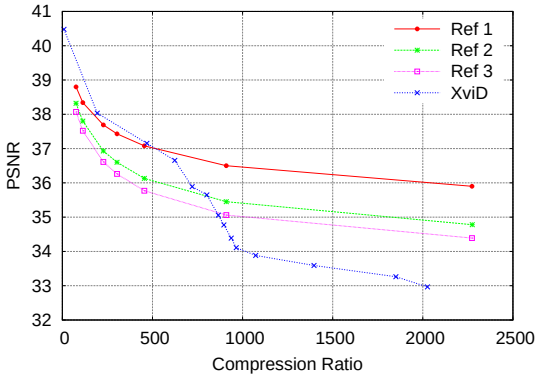
Figure 4.27: Ballet Stream 3 Color Quantizer 5



(a) Depth Quantizer 5

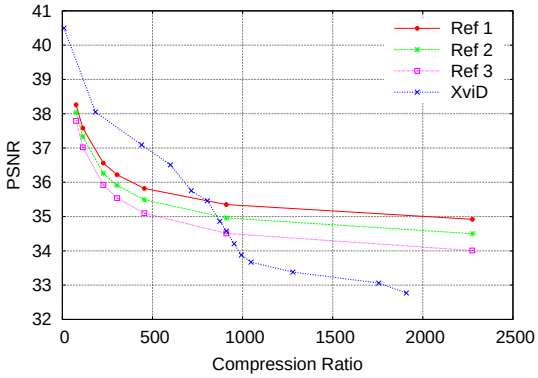


(b) Depth Quantizer 15

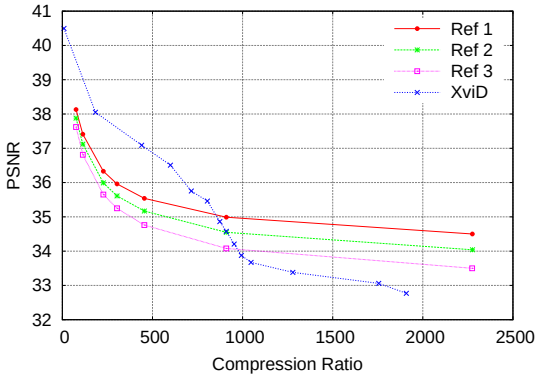


(c) Depth Quantizer 25

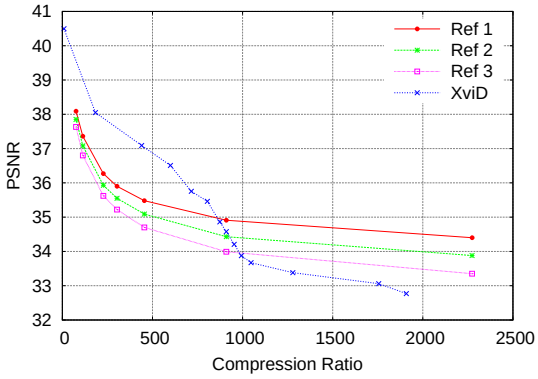
Figure 4.28: Ballet Stream 4 Color Quantizer 5



(a) Depth Quantizer 5

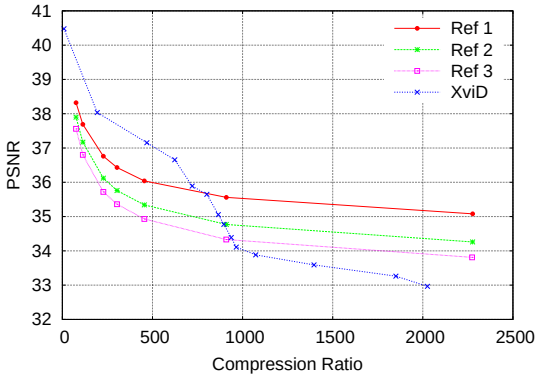


(b) Depth Quantizer 15

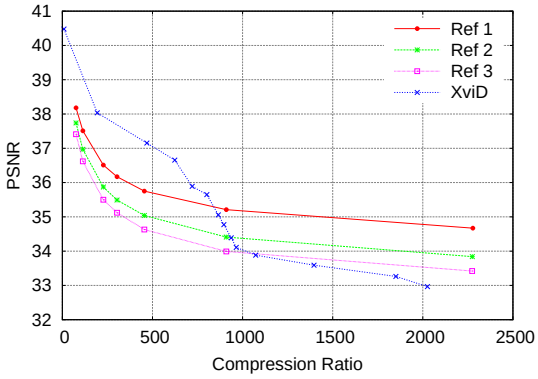


(c) Depth Quantizer 25

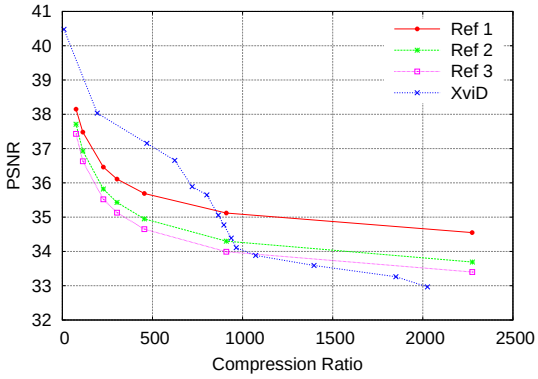
Figure 4.29: Ballet Stream 3 Color Quantizer 15



(a) Depth Quantizer 5

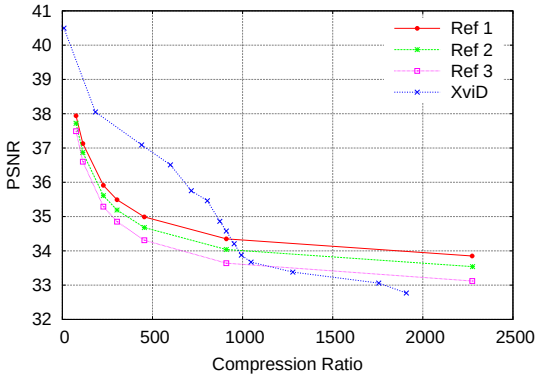


(b) Depth Quantizer 15

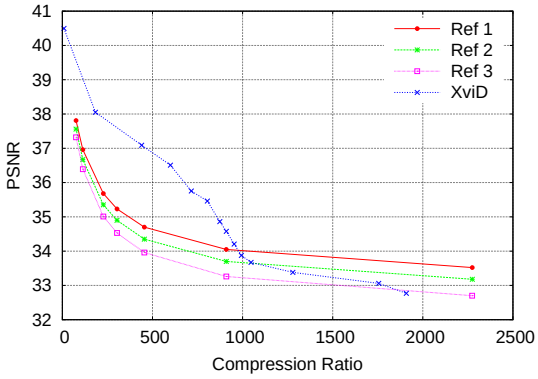


(c) Depth Quantizer 25

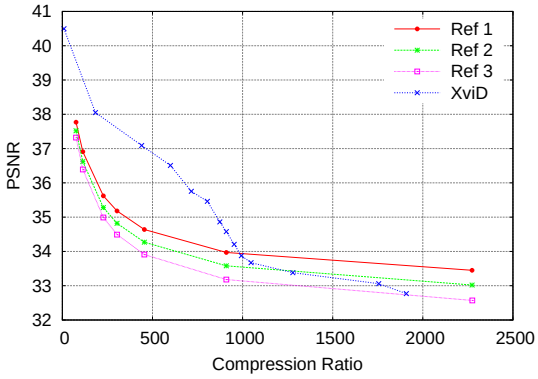
Figure 4.30: Ballet Stream 4 Color Quantizer 15



(a) Depth Quantizer 5

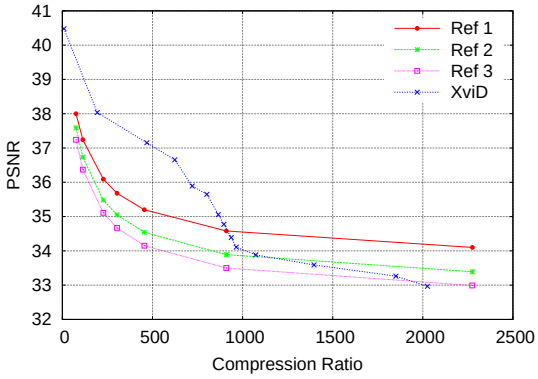


(b) Depth Quantizer 15

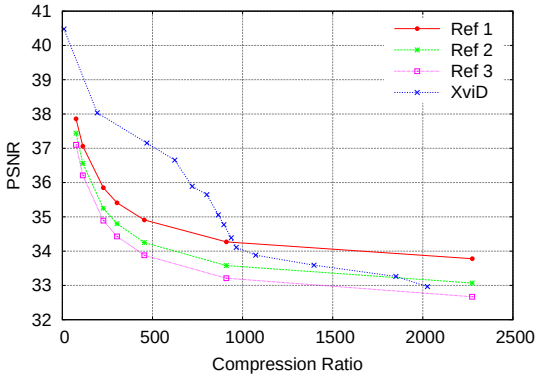


(c) Depth Quantizer 25

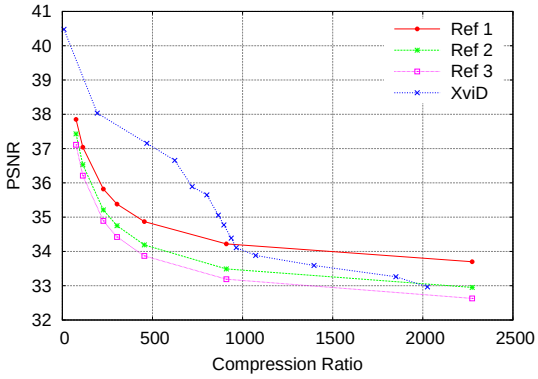
Figure 4.31: Ballet Stream 3 Color Quantizer 25



(a) Depth Quantizer 5



(b) Depth Quantizer 15



(c) Depth Quantizer 25

Figure 4.32: Ballet Stream 4 Color Quantizer 25

Figure 4.29b, Figure 4.30b, Figure 4.31b, and Figure 4.32b show the encoding results for different depth quantizers for all frames set to 15. Figure 4.27c, Figure 4.28c, Figure 4.29c, Figure 4.30c, Figure 4.31c, and Figure 4.32c show the encoding results for different depth quantizers for all frames set to 25. For all figures, the X-axis shows the compression ratio, and the Y-axis shows the PSNR of the decoded frame. Also, *Ref 1* shows the result for when the nearest streams on each side were selected as spatial reference streams, *Ref 2* shows the result for using the second nearest streams on each side as spatial reference streams, *Ref 3* shows the result for using the third nearest streams on each side as spatial reference streams, and *XviD* shows the result for encoding the stream using the XviD codec. As with the Breakdancers results, all XviD encoding results in Figure 4.27, Figure 4.29, and Figure 4.31, and all XviD encoding results in Figure 4.28, Figure 4.30, and Figure 4.32 are the same since the XviD encoding is independent of reference color quality, depth quality, or spatial reference streams.

In the Ballet data, the P-Stream encoding does not perform well compared to the XviD encoding. Even at high quality reference color (color quantizer 5) and high quality depth (depth quantizer 5), the P-Stream encodings are only comparable, and at high compression ratios P-Stream encoding is not significantly better than the XviD encoding (Figure 4.27a, Figure 4.28a). Also, XviD encoding even has higher PSNR at high compression ratio when the depth quality is low (Figure 4.31, Figure 4.32). This is due to the fact that Breakdancers data has faster motion than Ballet data, which leads to better temporal prediction in the Ballet data. This also results in significantly higher XviD encoding compression ratio for the Ballet data than the Breakdancers data.

4.4 Conclusion

For encoding multiple depth streams, the spatial coherence between streams can be utilized to efficiently encode the streams. In this chapter, algorithms for encoding P-Streams incorporating the spatial coherence between stream have been introduced and evaluated. In particular, the following have been presented:

- Three different algorithms – Temporal Reference Encoding, Spatial Reference Encoding, and Spatial and Temporal Reference Encoding – have been examined and evaluated for

encoding P-Stream depth. The results show that Temporal Reference Encoding is the reasonable choice.

- Evaluation of discrete cosine transform (DCT) based codec and discrete wavelet transform (DWT) based codec for encoding residuals. The results showed that the DWT based codec to be more desirable.

- An algorithm to effectively encode P-Stream color using spatial coherence between streams as well as the temporal coherence between frames. By utilizing the available depth information, the algorithm predicts a reference color on a pixel basis without using any motion compensation. It is also resilient enough to handle approximate depth values due to noise in the original data and lossy encoding.

- The reference mask was introduced to represent the source of color prediction per pixel, and four primitives to effectively encode the reference mask were described. The four primitives are the consensus filter, the delta depth threshold, a list of boxes, and a list of pixels.

- A comparison of the new P-Stream encoding algorithm with an MPEG based codec, which showed that the new algorithm performs well for scenes with fast motion. It also showed that the new algorithm can achieve higher compression ratios than the traditional codecs based on motion compensation.

Chapter 5

Reference Stream Selection

To encode multiple depth streams using spatial coherence between streams, reference streams must be selected. A *reference stream* serves as the basis stream for spatial prediction when encoding inter-streams. In Chapter 4, it was shown that the selection of reference streams affects encoding efficiency of inter-streams. However, most current research on multiple stream compression focuses on encoding individual streams – i.e. intra-stream and inter-stream encoding – only once a set of reference streams has been selected, rather than the problem of selecting reference streams from a set of streams in the first place. Often a user is required to pre-select reasonable streams as reference streams. This may be plausible when only a few streams exist, but as the number of streams increase the task becomes more challenging and an automatic approach is called for. The reference stream selection algorithm that I present in this chapter is an extension of an algorithm I originally presented in [Kum and Mayer-Patel 2005]. I am unaware of any other related research on the problem of reference stream selection.

In this chapter, algorithms for selecting reference streams are examined. First, in Section 5.1, the issues encountered when encoding reference streams are discussed. Then in Section 5.2, two different algorithms for reference stream selection are presented. The two algorithms presented in Section 5.2 are evaluated in Section 5.3, and conclusions are presented in Section 5.4.

5.1 Reference Stream Encoding

As presented in previous chapters, for multiple depth stream encoding using spatial coherence, a stream can be encoded as one of two different types. One type is the *intra-stream (I-Stream)*. An I-Stream is encoded without referring to any other streams, so it can be decoded independently. The other type is the *inter-stream (P-Stream)* that uses spatial coherence information from other streams (reference streams) to improve encoding efficiency. Therefore, P-Streams cannot be decoded without initially decoding associated reference streams. This is very similar to encoding frames as I-Frames and P-Frames using motion compensation. Since all reference frames are from the past and have already been decoded, P-Frames can use I-Frames or P-Frames as reference frames without incurring any decoding latency. However, for stream encoding, since the spatial reference frames from reference streams for decoding P-Streams are from the same instance in time, there is a latency penalty associated with decoding P-Streams. This penalty arises because the current frame of the reference stream – i.e. spatial reference frame – must be decoded before the current frame of the P-Stream can be decoded. Furthermore, if the reference stream itself had been encoded as a P-Stream, the latency is further increased as two spatial reference frames need to be decoded – the reference stream and the reference stream of the reference stream. As the chain of dependent reference streams encoded as P-Streams grows, the decoding latency increases. Therefore, for multiple depth stream encoding algorithm presented in this thesis, reference streams are always encoded as I-Streams, and non-reference streams as P-Streams. This limits the latency penalty to a maximum of decoding one frame.

Another restriction for encoding multiple streams using multiple reference streams is network bandwidth. The data acquisition system for systems that employ multiple streams generally use multiple cameras connected to multiple computers. Therefore, reference streams must be distributed over the local network to any computing hosts responsible for encoding the P-Streams. Multicast can be employed to make this distribution efficient. However, though the local network bandwidth is large, it is not infinite. This imposes a limit on the number of streams that can be designated as reference streams.

When encoding multiple depth streams, spatial coherence between streams is exploited by

projecting pixels from the current stream into the reference stream. Using depth values, each pixel of a P-Stream can be projected into the reference stream to find the corresponding pixel. The color of the corresponding pixel is then used as the reference color for the encoding pixel and only the *residual* – difference between the reference color and the color of the encoding pixel – need to be encoded.

One of the following three situations will arise when projecting pixels:

- The projection produces a similar reference color – the optimal situation since the residual will be very small.
- The pixel will project outside the reference stream (referred to as a *peripheral pixel*). Peripheral pixels occur because the current stream and the reference stream do not acquire the environment from the same viewpoint. This results in sections of the current stream that are not visible in the reference stream. Therefore a peripheral pixel does not have a spatial reference color to use as a basis for prediction.
- The projection could generate an inaccurate reference color. This results in large residuals which reduces the efficiency of the encoding.

Inaccurate spatial reference colors can occur when the depth value used to project a pixel is inaccurate, resulting in an incorrect corresponding pixel and reference color. An inaccurate depth value usually occurs around object borders where multiple objects are visible in a pixel. Thus, the depth value for the border pixel is an average of the depth values of the objects.

An inaccurate reference color can also occur due to different sampling rates of the reference stream and the current stream. As the reference stream and the current stream are not acquired from the same view, objects do not occupy the same area in the frame for both views. A good example is a poster acquired from the front in the current stream (i.e., the target stream to be encoded as an an I-Stream) and from an acute angle in the reference stream at the same distance. In this example, a pixel of the poster in the reference stream represents a larger area than a pixel of the poster in the current stream. So for a pixel in the reference stream that corresponds to high frequency content areas on the poster, the high frequency content will be filtered. When a pixel in this area is projected into the reference stream, the reference color

will not be correct.

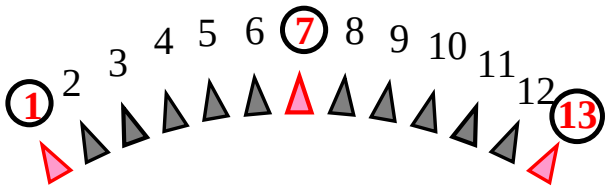
A third cause of inaccurate reference color can be due to occlusion. A surface may be occluded from view in the reference stream but not in the current stream. If a pixel from this surface in the current stream (referred to as an *occluded pixel*) is projected into the reference stream, the corresponding pixel color will be from a different surface. While it is difficult to detect the previous two causes of inaccurate reference colors, detecting occluded pixels is possible using depth. If the projected pixel depth does not match the depth of the corresponding pixel in the reference stream, the pixel can be identified as an occluded pixel. Like peripheral pixels, occluded pixels will not have any spatial reference color to use as a basis for prediction.

When multiple reference streams are used for encoding, a pixel may have multiple reference colors. In such instances, depth can be used to select one reference color by using the reference color from the reference pixel with the smallest difference in depth to the projected target pixel. Furthermore, a pixel may have no spatial reference colors – pixels that are either peripheral pixels or occluded pixels in all reference streams. These pixels can use temporal prediction for its reference color – the color from the previous frame at the same pixel location.

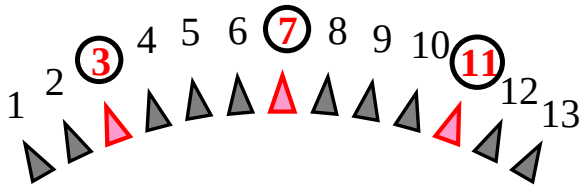
5.2 Reference Stream Selection

There are two main approaches to selecting reference streams. One approach is to select reference streams which represent the most volume of the scene ($RefStr_c$). The other is to select reference streams such that the volume overlap between the reference streams and the non-reference streams is the maximum ($RefStr_o$).

Selecting reference streams to maximize volume coverage ($RefStr_c$) emphasizes reducing peripheral pixels. Therefore, for a linear configuration of the streams, the streams at both ends are first selected as reference streams. Then for selecting n reference streams, the space between the initial reference streams is divided equally into $n-1$ sections. Streams that are closest to the section borders are then selected as reference streams. Figure 5.1a shows an example of selecting three reference streams from a total of thirteen streams that maximizes volume coverage. For more complex stream configurations, selecting reference streams to maximize



(a) Maximum Coverage ($RefStr_c$)



(b) Maximum Overlap Between Reference Streams and Non-Reference Streams ($RefStr_o$)

Figure 5.1: Example of Reference Stream Selection (Reference Streams Circled)

volume coverage is not trivial.

Selecting reference streams to maximize overlap between reference streams and non-reference streams ($RefStr_o$) emphasizes on reducing occluded pixels. However, compared to $RefStr_c$, the number of peripheral pixels increases due to decrease in total volume coverage. Except for some trivial stream configurations, it is difficult to select reference streams to maximize volume overlap between reference streams and non-reference streams. Therefore, the problem is simplified to selecting reference streams that maximize the overlap between non-reference streams and its closest reference stream. This simplification is made under the observation that the effect of the closest spatial reference stream is significantly larger than the subsequent spatial reference streams.

The volume overlap between two streams is expensive to calculate, therefore, the angle between the view directions of two streams is used to approximate volume overlap. Empirically, the view directions of two streams are a good estimate for how much the two stream volumes overlap – smaller the angle, bigger the overlap. Figure 5.1b shows an example of selecting three reference streams from a total of thirteen streams that maximizes volume overlap of reference streams with non-reference streams.

The group based approach presented in [Kum et al. 2003] for finding reference streams in

groups can be used for selecting reference streams to maximize volume overlap. However, some modifications are required due to characteristic differences. In [Kum et al. 2003], the reference streams change dynamically each frame. Furthermore, the reference streams are used to remove redundant points – i.e. reference streams are needed for only encoding a stream but not for decoding. Therefore, reference streams can be encoded as P-Streams without introducing any decoding latencies; only one reference stream is encoded as an I-Stream while all other streams, reference and non-reference streams are encoded as P-Streams. Consequently, the volume overlap between reference streams affects encoding efficiency. However, for encoding multiple depth streams, all reference streams are encoded as I-Streams to reduce decoding latency. Therefore, only the overlap between reference streams and non-reference streams affects encoding efficiency.

In the following sections, an algorithm for reference stream selection to maximize volume overlap between reference streams and non-reference streams ($RefStr_o$) is presented. In Section 5.2.1, the effective criteria for partitioning n streams into k groups and for selecting the reference stream in each group is introduced. These metrics are utilized to partition streams and to select reference streams in Section 5.2.2. In Section 5.2.3, the metrics are used to develop an efficient approximate algorithm for stream partitioning and reference stream selection when n is too large for an exhaustive approach.

5.2.1 Metrics

As previously mentioned, exact calculation of the volume overlap between two streams is expensive. Therefore, the angle between the view directions of two streams is used to approximate stream volume overlap. Using the angle between stream views, the streams are divided into groups and the stream that best represents each group is selected as the reference stream. Hence the metrics used to divide streams into groups and selecting the reference streams are based on the angle between stream views. The three metrics used for reference stream selection are *local squared angle sum*, *group squared angle sum*, and *total squared angle sum*.

The *local squared angle sum* (LSAS) is defined for stream S_i as the sum of the squared angle between stream S_i and all other streams in its group. If stream S_i and S_j is in group k,

and n_k is the number of streams in group k, LSAS for stream S_i (LSAS_i) can be calculated using Equation 5.1. This criterion is used for selecting the reference stream of a group. The stream with the lowest LSAS in the group is chosen as the reference stream of the group.

$$\text{LSAS}_i = \sum_{j=1}^{n_k} [\text{angle}(S_i, S_j)]^2 \tag{5.1}$$

The *group squared angle sum* (GSAS), defined for a given group, is the sum of the squared angle between the group's reference stream and every other stream in the group. If R_j is the reference stream in group j, S_{ji} is a non-reference stream in group j, and n_j is the number of non-reference streams in group j, GSAS for group j (GSAS_j) can be formulated as Equation 5.2. This criterion is used to evaluate partitions of n streams into k groups. Note that GSAS_j is same as the LSAS of stream R_j.

$$\text{GSAS}_j = \sum_{i=1}^{n_j} [\text{angle}(R_j, S_{ji})]^2 \tag{5.2}$$

The sum of all GSAS for a particular stream partition is defined as the *total squared angle sum* (TSAS). TSAS is calculated using Equation 5.3, where k is the number of groups. The partition with the minimum TSAS is selected as the optimal solution.

$$\text{TSAS} = \sum_{i=1}^{k} GSAS_i \tag{5.3}$$

5.2.2 Exhaustive Selection

One way to select k reference streams from n streams is to do an exhaustive search – i.e. examine all possible combinations, a total of ${}_nC_k$.

The steps for the exhaustive reference stream selection algorithm are the following:

1. Select k streams from n streams as reference streams. Assign each selected reference stream as the reference stream of a group.

2. Assign all other streams to a group. The streams are assigned to the group where the absolute angle of the stream and the group's reference stream is the smallest.

3. Calculate TSAS for this selection.

4. Repeat steps 1 - 3 for all possible combinations of stream selection ($_nC_k$).

5. Choose the stream selection with the smallest TSAS as the reference streams.

Unless n is small, this method is not practical.

5.2.3 Approximate Selection

The k-means framework ([Jain et al. 1999]), originally developed as a clustering algorithm, can be used to select reference streams for an approximate solution. The k-means framework is used to partition n data points into k disjoint subsets such that a criterion is optimized. The k-means framework is a robust and fast iterative method that finds the locally optimal solutions for the given criteria. The k-means algorithm involves the following three steps:

1. Initialization: Initial centers for the k partitions are chosen.

2. Assignment: All data points are placed in the partition with the center that best satisfies the given criteria. Usually the criteria are given as a relationship between a data point and the center.

3. Centering: For each partition the cluster centers are reassigned to optimize the criteria.

The assignment and centering steps are repeated until the cluster centers do not change or an error threshold is reached.

Iterative Solution for Approximate Selection

The steps for the iterative solution for approximate reference stream selection are as follows:

1. Generate initial reference streams.

2. Designate the initial reference streams as the reference stream of each group.

3. Assign all other streams to a group. The streams are assigned to the group where the absolute angle of the stream and the reference stream is the smallest.

4. Compute a new reference stream for each group. The stream with the smallest LSAS in the group is designated as the new reference stream of the group.

5. Repeat steps 3 - 4 until the reference streams converge and do not change between iterations.

Reference Stream Initialization

The performance of this approximate approach heavily depends on the initial starting conditions – initial reference streams and stream order ([Peña et al. 1999]). Therefore in practice, to obtain a near optimal solution, multiple trials are attempted with several different instances of initial reference streams. The TSAS of the resulting solution is calculated and the solution with the smallest TSAS is selected.

The full search space for the iterative method could be investigated when all possible starting conditions – total of ${}_nC_k$ – are explored. A starting condition is given as a set of k initial reference streams. As seen in Table 5.1, such an exhaustive method will find all possible ending conditions. For example, if $n = 10$ and $k = 5$, there would be a total of ${}_{10}C_5 = 252$ possible starting conditions, which for the example of Table 5.1a leads to one of 46 distinct ending conditions. An ending condition is one in which the reference streams do not change from one iteration to the next.

However, for numbers of n and k where this becomes impractical, only a small sample of all possible starting conditions can be examined. The chances of finding the same optimal solution as the exhaustive method will increase by intelligently selecting the starting conditions to examine.

Theoretically, the best way to initialize the starting reference streams is to have the initial reference streams be as close to the optimal answer as possible. This means that in general the initial reference streams should be dispersed throughout the data space – i.e. the angles between the initial reference streams should be as large as possible. In this section, a method for finding initial reference streams that are well dispersed in the data space is presented. Once good starting conditions have been identified, it is straightforward to apply the approximate reference stream selection algorithm and examine TSAS of all corresponding ending conditions to find a near-optimal solution.

The basic approach to identifying good starting conditions is the following:

1. Sort the given streams (Stream Sorting).

2. Group the streams in all possible *reasonable* ways (Initial Group Partitions).

3. Find all possible *reasonable* candidate reference streams (Reference Stream Candidates).

4. Generate all possible combinations of the candidate reference streams for each possible grouping as a starting condition with all duplicates removed (Set of Initial Reference Streams).

Stream Sorting Although the streams cannot be strictly ordered due to the three-dimensional nature of the viewpoint locations, an approximate sorting using the global squared angle sum should be sufficient. *Global squared angle sum* (GlSAS) for stream S_i is the sum of the squared angle between steam S_i and all other given streams. Equation 5.4 shows how to calculated GlSAS for stream S_i (GlSAS_i), where S_i and S_j are streams, and n is the total number of streams.

$$\text{GlSAS}_i = \sum_{j=1}^{n} [\text{angle}(S_i, S_j)]^2 \tag{5.4}$$

Given n streams, the *pivot stream* is chosen as the stream with the lowest GlSAS. Next all other streams are divided into three groups – streams with a negative angle with the pivot stream, streams with a positive angle with the pivot stream, and streams with zero angle with the pivot stream.

Any stream that has zero angle with the pivot stream, either covers the pivot stream or is covered by the pivot stream. All such streams can be represented by one stream, the *dominant stream*. A dominant stream is a stream that overlaps all other streams. Objects present in the non-dominant streams should be present in the dominant stream. Therefore, for stream partitioning, the dominant stream can be used to represent the non-dominant streams. The non-dominant streams are removed from the stream list and added back as non-reference streams after the reference streams has been selected. Figure 5.2 illustrates the notion of a dominant stream. Streams S_1, S_2, and S_3 all have zero angle with each other. Stream S_1 overlaps stream S_2 and S_3 since any object present in S_2 and S_3 is also present in S_1.

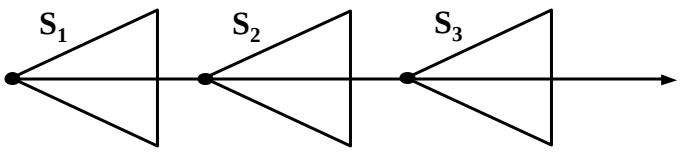


Figure 5.2: Dominant Stream S_1

Therefore, S_1 is the dominant stream. The arrow indicates the stream view direction.

The positive angle and negative angle groups are each sorted using the GlSAS. The negative angle streams are sorted in descending order and the positive angle streams are sorted in ascending order. Placing the sorted negative angle streams on the left and the sorted positive angle streams on the right of the pivot stream creates the final sorted list.

Initial Group Partitions After the n streams are sorted, they are partitioned into k initial groups. To ensure that all possible good starting conditions are included, all reasonable groupings are considered.

The k initial groups are created by the following reasonableness criteria:

1. If stream S_i is in group G_k, then all streams right of stream S_i in the sorted stream list are in group G_k.

2. If stream S_i is in group G_j, stream S_{i+1} is either in group G_j or G_{j+1} where $1 \leq i < n$, $1 \leq j < k$, and stream S_i is left of stream S_{i+1} in the sorted stream list.

3. If n is not an exact multiple of k, every group is assigned either $\lfloor \frac{n}{k} \rfloor$ or $\lceil \frac{n}{k} \rceil$ streams and every stream is assigned to a group. If n is an exact multiple of k one group is assigned $\frac{n}{k} - 1$ streams, another is assigned $\frac{n}{k} + 1$ streams, and every other group is assigned $\frac{n}{k}$ streams.

Streams are grouped into every possible combination that meets the above reasonableness criteria. Examples are shown in Figure 5.3. Each number represents a stream and the stream numbers indicated the sorted order. Also, the streams in the same column belong to the same group, while each row is a particular stream grouping partition for the given number of streams and groups. Figure 5.3a shows different methods of creating three groups ($n = 3$) from ten sorted streams ($k = 10$). Since there are ten streams to be partitioned into three groups, a

Group 1	Group 2	Group 3
1, 2, 3	4, 5, 6	7, 8, 9, 10
1, 2, 3	4, 5, 6, 7	8, 9, 10
1, 2, 3, 4	5, 6, 7	8, 9, 10

(a) All possible initial group partitions for 10 streams into 3 groups ($k = 10, n = 3$)

Group 1	Group 2	Group 3
1, 2, 3	4, 5, 6	7, 8, 9

(b) Only one grouping is possible for 9 streams and 3 groups for normal conditions ($k = 9, n = 3$)

Group 1	Group 2	Group 3
1, 2	3, 4, 6	6, 7, 8, 9
1, 2	3, 4, 5, 6	7, 8, 9
1, 2, 3	4, 5	6, 7, 8, 9
1, 2, 3	4, 5, 6, 7	8, 9
1, 2, 3, 4	5, 6, 7	8, 9
1, 2, 3, 4	5, 6,	7, 8, 9

(c) All possible initial group partitions for 9 streams into 3 groups ($k = 9, n = 3$)

Figure 5.3: Initial Group Partitions

group must have either three streams ($\lfloor \frac{10}{3} \rfloor$) or four streams ($\lceil \frac{10}{3} \rceil$) – criteria 3. Therefore, one group must have four streams and two groups must have three streams. Additionally, each group must also maintain the sorted order (criteria 1 and 2), so only three initial group partitions are possible.

The special case when n is an exact multiple of k is treated differently because the normal conditions will only allow one possible partition – all groups with $\frac{n}{k}$ number of streams. Figure 5.3b shows the only possible groupings of nine sorted streams ($k = 9$) into three groups ($n = 3$) without the special case exception. Initial starting conditions are critical for a good solution to the approximate stream partition, so generating multiple starting conditions is desirable. To generate multiple partitions for the special case, one group is assigned $\frac{n}{k} - 1$ streams, another is assigned $\frac{n}{k} + 1$ streams, and every other group is assigned $\frac{n}{k}$ streams. For the example of grouping nine streams into three groups, one group has two streams ($\frac{9}{3} - 1$), another group has four streams ($\frac{9}{3} + 1$), and all other groups have three streams ($\frac{9}{3}$). For such a partition, while maintaining sorted order, six different combinations of stream partitioning is possible (Figure 5.3c).

Group 1	Group 2	Group 3
1, 2, 3	4, 5, 6	7, 8, 9, 10

Figure 5.4: Reference Stream Candidates Underlined for 10 Streams Partitioned into 3 Groups

Reference Stream Candidates Again to ensure all possible good starting conditions are included, multiple reference stream candidates are chosen for each group. If the group has even number of streams, the two streams in the middle are selected as candidates. If it has odd number of streams, the middle stream and its two neighboring streams are chosen as candidates. Figure 5.4 shows one of the group partitions from Figure 5.3a with the candidate reference streams underlined. Streams 1, 2, and 3 are candidate references streams for Group 1; Stream 4, 5, and 6 are candidate references streams for Group 2; and Stream 8 and 9 are candidate references streams for Group 3.

Set of Initial Reference Streams Finally, a set of k reference streams are selected, one from each group, to construct a starting condition – the initial reference streams. All possible combinations for the candidate reference streams are generated as good starting conditions. Figure 5.5 is a list of all possible starting conditions – i.e. initial reference streams – that can be generated from reference stream candidates in Figure 5.4. Note that there can be duplicate starting conditions created, for which all but one are removed. The distinct starting conditions generated will be the good starting conditions (initial reference streams) explored.

If the number of starting conditions is still too large, the desired number of initial sets can be stochastically sampled from all identified good starting conditions. To provide starting conditions with duplicates a better chance of being chosen, the duplicates are not removed when stochastically sampling to generate initial reference streams.

{1, 4, 8}, {1, 4, 9}, {1, 5, 8}, {1, 5, 9}, {1, 6, 8}, {1, 6, 9},
{2, 4, 8}, {2, 4, 9}, {2, 5, 8}, {2, 5, 9}, {2, 6, 8}, {2, 6, 9},
{3, 4, 8}, {3, 4, 9}, {3, 5, 8}, {3, 5, 9}, {3, 6, 8}, {3, 6, 9}

Figure 5.5: Set of Initial Reference Stream Candidates from Figure 5.4

(a) **Breakdancers Color**

(b) **Breakdancers Depth**

(c) **Ballet Color**

(d) **Ballet Depth**

Figure 5.6: Frame from Breakdancers and Ballet

5.3 Results

In this section, the reference stream selection stream algorithm is evaluated. First, in Section 5.3.1, the effectiveness of approximate reference stream selection algorithm is compared to the exhaustive method. Then the two different reference stream selection methods – maximizing volume overlap between reference streams and non-reference streams ($RefStr_o$), and maximizing volume coverage of reference streams ($RefStr_c$) – are compared using the Ballet and Breakdancers, the two data sets from [Zitnick et al. 2004]. Both data sets have eight depth streams which are 100 frames long at 1024×768 resolution and was captured at 15 fps. Each frame has 24 bit color (RGB) and 8 bit depth information for each pixel. The depth was computed using the technique described in [Zitnick et al. 2004]. Example frames from both data sets are shown in Figure 5.6.

Total number of streams & reference streams	$n = 10, k = 5$	$n = 22, k = 5$	$n = 25, k = 5$	$n = 60, k = 5$
Number of possible initial reference stream sets	252	26334	53130	5461512
Number of created initial reference stream sets	144	240	529	352
Total number of resulting reference stream sets	46	10443	25907	5170194
Optimal reference streams	{2, 3, 6, 8, 9}	{4, 15, 16, 17, 20}	{3, 11, 12, 13, 17}	{2, 8, 22, 34, 58}
TSAS of the optimal reference streams	246	574.25	686.5	2224
Resulting reference streams	{2, 3, 6, 8, 9}	{9, 15, 16, 17, 20}	{3, 11, 12, 13, 21}	{22, 34, 36, 47, 54}
TSAS of resulting reference streams	246	734	730.5	3984.5
Number of reference stream sets with less TSAS	0	6	1	20517
	(a)	(b)	(c)	(d)

Table 5.1: Reference Stream Selection

5.3.1 Reference Stream Selection

Since the Ballet and Breakdancers data set had only one stream configuration for each data set with only 8 streams, different stream setups were randomly generated to compare the approximate reference stream selection algorithm with the exhaustive method. For each generated stream setup, all streams were placed randomly with the only constraint that the largest possible angle between two stream views was 120 degrees.

Table 5.1 shows the result of running the reference stream selection algorithm on several different examples. In row *Total number of streams & reference streams*, n is the total number of streams and k is the number of reference streams. Row *Number of possible initial reference stream sets* shows the total number of possible initial reference stream sets for the given example – ${}_nC_k$. Row *Number of created initial reference stream sets* is the total number of possible initial reference stream sets created using the algorithm given in Section 5.2.3. Row *Total number of resulting reference stream sets* is the total number of possible reference stream sets resulting from running the approximate reference stream selection algorithm (Section 5.2.3) on the created initial reference stream sets. Row *Optimal reference streams* shows the reference streams selected from all possible results generated by an exhaustive search – reference streams with the smallest total squared angle sum (TSAS). Row *TSAS of the optimal reference streams* is the TSAS of the reference streams in row *Optimal reference streams*. Row *Resulting reference streams* shows the reference streams selected by the approximate reference streams selection algorithm from the created initial reference stream sets – i.e. from all of the resulting reference stream sets from the exhaustive method, the reference stream set with the smallest TSAS. Row *TSAS of resulting reference streams* is the TSAS value of the reference streams in row *Resulting reference streams*. Row *Number of reference stream sets with less TSAS* is the number of reference stream sets from row *Total number of resulting reference stream sets* that has less TSAS than the reference streams in row *Resulting reference streams*.

Table 5.1a ($n = 10, k = 5$) indicate that for a relatively small n and k an exhaustive search is plausible. Table 5.1c ($n = 25, k = 5$) is an example of where n is an exact multiple of k. Table 5.1d ($n = 60, k = 5$) confirms doing an exhaustive search is not practical for large number of streams – total of 5461512 (${}_{60}C_5$) possible reference streams selection must

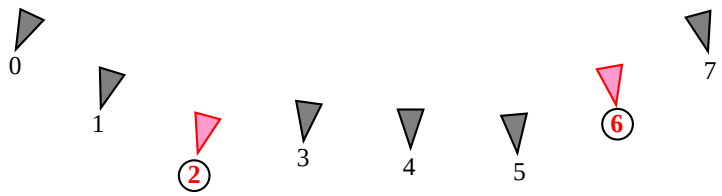
	Reference Streams	TSAS	Initial Reference Stream Sets
Breakdancers	2, 6	145.840	28
	0, 7	332.880	N/A
Ballet	1, 5	187.655	11
	2, 5	181.250	17
	0, 7	416.502	N/A

Table 5.2: Reference Stream Selection for Breakdancers and Ballet

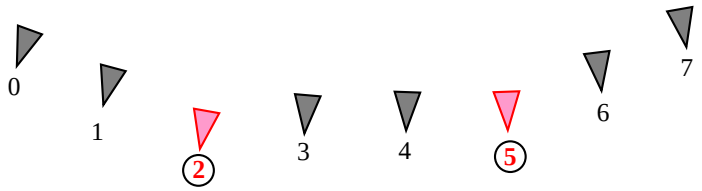
be examined. However, using the approximate reference streams selection algorithm, only 352 initial number of reference streams were examined – i.e. approximately 0.006% of all possible reference streams. The selected reference streams ({22, 34, 36, 47, 54}) were in the top 0.4% of all possible reference stream sets. Table 5.1 demonstrates the approximate reference streams selection algorithm works well for different number of streams. For all examples, the reference streams selected with the approximate reference streams selection algorithm is either the same or very close to the optimal solution. All reference streams selected with the approximate method were in the top 0.4% of the all possible reference stream sets. This was achieved with exploring less than 1% of the total possible initial reference stream sets when the number of streams was larger than 22.

For Breakdancers and Ballet, two reference streams were selected because there were total of eight streams. Furthermore, since there were only eight streams, all 28 ($_8C_2$) possible combinations were tried as initial reference streams. Table 5.2 shows the results from reference streams selected for maximizing volume overlap between reference streams and non-reference streams ($RefStr_o$), and reference streams for maximizing volume coverage ($RefStr_c$) – streams 0 and 7. The column *Reference Streams* shows the two selected reference streams. The column *TSAS* shows the TSAS value for the corresponding reference streams. The column *Initial Reference Stream Sets* shows the number initial reference stream sets that resulted in the specified reference streams from the reference streams selection algorithm.

For Breakdancers data, all 28 possible starting reference stream sets resulted in one possible reference stream selection – streams 2 and 6. For Ballet data, two possible reference stream selections were generated from 28 initial reference stream sets selection and the one with the smaller TSAS was selected – streams 2 and 5. This also had more initial starting points.



(a) Breakdancers



(b) Ballet

Figure 5.7: Streams of Breakdancers and Ballet with Reference Streams Circled

Figure 5.7 shows the stream configuration for both data with the two reference streams selected for maximizing volume overlap between reference streams and non-reference streams ($RefStr_o$) circled.

5.3.2 Predicted Image

The two different reference stream selection methods – maximizing volume overlap between reference streams and non-reference streams ($RefStr_o$), and maximizing volume coverage of reference streams ($RefStr_c$) – are evaluated using two different metrics. One is the quality of the predicted image generated from the reference frames. The other is the compression ratio of the residuals. In this section, results for the predicted images are presented.

The predicted image is generated by projecting each pixel of the current frame into the spatial reference frames to find the corresponding value. When corresponding values exist in multiple reference streams, the closest one (smallest depth difference) is selected. However, there are pixels that do not have any correspondences – i.e. occluded and peripheral pixels. The colors for these occluded and peripheral pixels are temporally predicted – color of the pixel at the same location in the previous frame.

Figure 5.8 shows the predicted image for a Ballet frame from stream 3 at depth and color quantizer of 5. The reference streams selected were stream 2 and 5, and the spatial reference

(a) Reference Frame from Stream 2

(b) Reference Frame from Stream 5

(c) Predicted Image from Stream 2

(d) Predicted Image from Stream 5

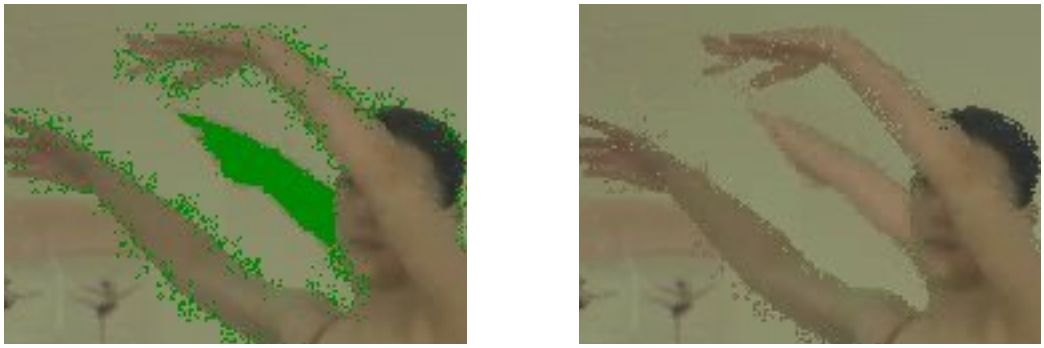
(e) Spatially Predicted Image

(f) Temporal Reference Frame

(g) Original Frame

(h) Predicted Image

Figure 5.8: Ballet Predicted Image for Stream 3 at Color and Depth Quantizer 5



(a) Spatial Prediction **(b) After Temporal Prediction**

Figure 5.9: Example of Inaccurate Temporal Prediction

frames from these reference streams are Figures 5.8a and b. Figure 5.8c is the projected spatial reference frame of Figure 5.8a. The occluded and peripheral pixels are shown in green. Similarly, Figure 5.8d is the projected spatial reference frame for Figure 5.8b. Figure 5.8e is the combined spatial reference image of Figures 5.8c and d. For pixels that have reference colors in both frames, the closer one was chosen. Figure 5.8f is the previous frame from stream 3. Figure 5.8g is the current frame to be encoded. Figure 5.8h is final predicted image where occluded and peripheral pixels in Figure 5.8e used temporal prediction (Figure 5.8f) for its color.

The predicted image (Figure 5.8h) is a good approximate of the current frame (Figure 5.8g). An exception is the wall behind the ballerina's head. There seems to be a third arm between her two arms where the wall is in the original frame. This is an example of inaccurate temporal prediction. Figure 5.9a is a closeup of Figure 5.8e. The occluded and peripheral pixels, in green, near the head are mostly due to occlusion – the portion of the wall is occluded in stream 2 by her left arm, and in stream 5 by her right arm. Since no spatial predicted values are available, temporal prediction was used for the occluded pixels. Figure 5.9b is a closeup of Figure 5.8h which shows the occluded pixels with temporal prediction. Part of the occluded area of the wall is also not visible in the previous frame (Figure 5.9f) – the left arm is occluding the wall in the previous frame – resulting in inaccurate predicted values.

When using predicted images to evaluate the effectiveness of reference streams, careful consideration must be given. First, the object of the predicted image is to evaluate the reference streams for spatial prediction. Therefore, spatially predicted values should be selected when

possible, even if a better temporal reference value exists. Additionally, using temporal reference values for pixels that do not have spatial reference values may distort results. Consider a frame with significant number of peripheral pixels, but excellent temporal predictions for these peripheral pixels. If every pixel was considered in evaluating this predicted image, the image would be identified as excellent since most of the pixels have excellent reference values. However, since a significant number of the pixels were peripheral pixels that would indicate that the reference streams were not optimal. Conversely, if only the pixels with spatial references were considered, the predicted image with many occluded and peripheral pixels would not be penalized. Therefore, results for predicted images with occluded and peripheral pixels (labeled as *All Pixels*) and without occluded and peripheral pixels (labeled as *Spatial Reference Pixels*) are both examined.

All frames in this experiment were encoded as P-Frames with the previous frame encoded as an I-Frame. Furthermore, all frames of the reference streams and depth were encoded as I-Frames. This eliminates error propagation when a P-Frame is encoded using a P-Frame as reference. Additionally, the quantizer for all blocks in a frame was kept constant for the same components. This insures that the encoding is affected similarly across blocks within the same frame. Finally, the color quantizer for all streams were kept same, as well as the depth quantizer for all streams.

Occluded and Peripheral Pixels

The number of occluded and peripheral pixels in the predicted image should indicate how good the selected reference streams are for spatial prediction. The fewer occluded and peripheral pixels in the predicted image, the better the selected reference streams.

Figure 5.10 shows the portion of pixels in the predicted image that are either peripheral pixels or occluded pixels averaged over 99 frames for six non-reference streams. The X-axis represents the depth quantizer, while the Y-axis represent the percentage of occluded and peripheral pixels in the predicted image. The four plots in the figure are for Breakdancers with $RefStr_o$ (reference streams 2 and 5), Breakdancers with $RefStr_c$ (reference streams 0 and 7), Ballet with $RefStr_o$ (reference streams 2 and 6), and Ballet with $RefStr_c$ (reference streams 0 and 7).

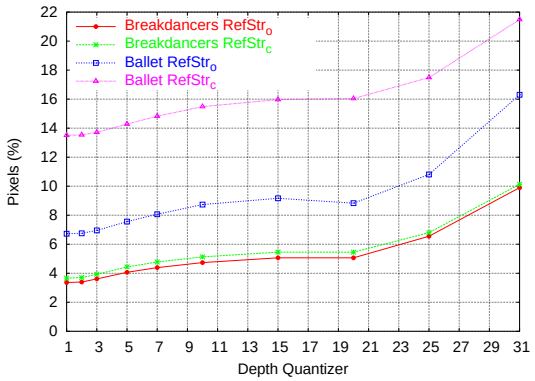


Figure 5.10: Occluded and Peripheral Pixels

It is apparent that the quality of depth affects the number of occluded and peripheral pixels – as the depth quality decreases the number of occluded and peripheral pixels increase. This is because depth quality affects the accuracy of depth, which influences projection of the pixel. Additionally, $RefStr_c$ results in more occluded and peripheral pixels than $RefStr_o$ which suggests $RefStr_o$ yields better reference streams.

Accumulative Residual Histogram

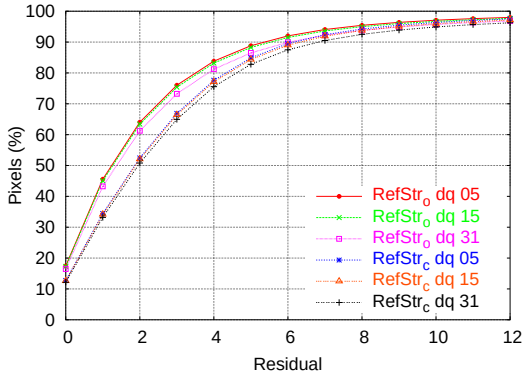
The *accumulative residual histogram*, which shows the number of pixels with residual equal to or less than a given value, is used to compare the effectiveness of the predicted images. The residual is defined as the difference between the current image and the predicted image. Predicted images that contain more pixels with smaller residuals are considered more effective since they generally require less bits to encode.

Figure 5.11 shows the average accumulative residual histogram of Breakdancers for *All Pixels* – predicted images with occluded and peripheral pixels. Figure 5.11a shows the accumulative residual histogram for color quantizer 5. The X-axis represents the residual between the predicted image and the current frame. The Y-axis shows the percentage of pixels that have residuals equal or smaller than the corresponding value. The figure has six different plots: $RefStr_o$ (streams 2 and 6) with depth quantizer 5 (dq 05), 15 (dq 15), and 31 (dq 31); and $RefStr_c$ (streams 0 and 7) with depth quantizer 5 (dq 05), 15 (dq 15), and 31 (dq 31). Similarly, Figure 5.11b is the accumulative residual histogram for color quantizer 15 and Figure 5.11c is the accumulative residual histogram for color quantizer 31. Figure 5.11d shows accumula-

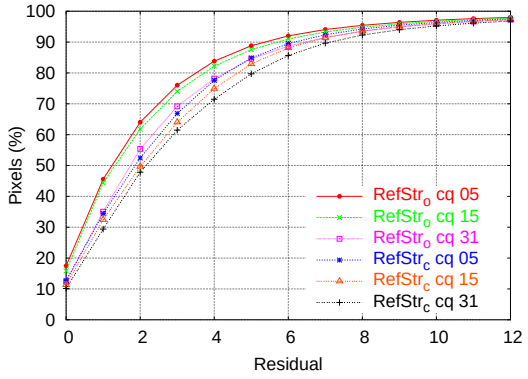
tive residual histogram for depth quantizer 5 – the plots are for $RefStr_o$ (streams 2 and 6) with color quantizer 5 (cq 05), 15 (cq 15), and 31 (cq 31); and $RefStr_c$ (streams 0 and 7) with color quantizer 5 (cq 05), 15 (cq 15), and 31 (cq 31). Similarly, Figure 5.11e is the accumulative residual histogram for depth quantizer 15 and Figure 5.11f is the accumulative residual histogram for depth quantizer 31. Depth quantizer seems to have little effect on the residuals (Figures 5.11a-c), while color quantizer does affect the residuals (Figures 5.11e-f). Furthermore, $RefStr_o$ is always significantly better than $RefStr_c$.

Figure 5.12 shows the average accumulative residual histogram of Breakdancers for *Spatial Reference Pixels* – predicted images without occluded and peripheral pixels. Figures 5.12a-c shows the accumulative residual histogram for color quantizer 5, 15, and 31. Each figure has six different plots: $RefStr_o$ (streams 2 and 6) with depth quantizer 5 (dq 05), 15 (dq 15), and 31 (dq 31); and $RefStr_c$ (streams 0 and 7) with depth quantizer 5 (dq 05), 15 (dq 15), and 31 (dq 31). Figure 5.12d-f shows accumulative residual histogram for depth quantizer 5, 15, and 31 with six plots each : $RefStr_o$ (streams 2 and 6) with color quantizer 5 (cq 05), 15 (cq 15), and 31 (cq 31); and $RefStr_c$ (streams 0 and 7) with color quantizer 5 (cq 05), 15 (cq 15), and 31 (cq 31). Compared to Figure 5.11, the total number of pixels does not reach 100% due to occluded and peripheral pixels. Furthermore, as depth quantizers increase, the number of occluded and peripheral pixels increase (Figures 5.12a-c). This is because as depth accuracy decreases, pixel projection error increases. As with *All Pixels*, comparing *Spatial Reference Pixels* for Breakdancers, $RefStr_o$ has more pixels with smaller residuals than $RefStr_c$.

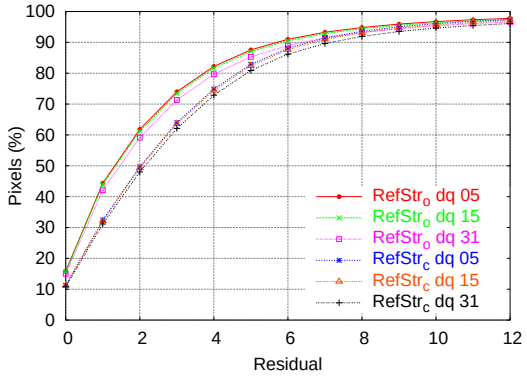
Figure 5.13 shows the average accumulative residual histogram of Ballet for *All Pixels*. Figures 5.13a-c shows the accumulative residual histogram for color quantizer 5, 15, and 31. Each figure has six different plots: $RefStr_o$ (streams 2 and 5) with depth quantizer 5 (dq 05), 15 (dq 15), and 31 (dq 31); and $RefStr_c$ (streams 0 and 7) with depth quantizer 5 (dq 05), 15 (dq 15), and 31 (dq 31). Figure 5.13d-f shows accumulative residual histogram for depth quantizer 5, 15, and 31 with six plots each : $RefStr_o$ (streams 2 and 5) with color quantizer 5 (cq 05), 15 (cq 15), and 31 (cq 31); and $RefStr_c$ (streams 0 and 7) with color quantizer 5 (cq 05), 15 (cq 15), and 31 (cq 31). Compared to Breakdancers (Figure 5.11), the difference between $RefStr_o$ and $RefStr_c$ is significantly less but $RefStr_o$ still has more pixels with smaller residuals.



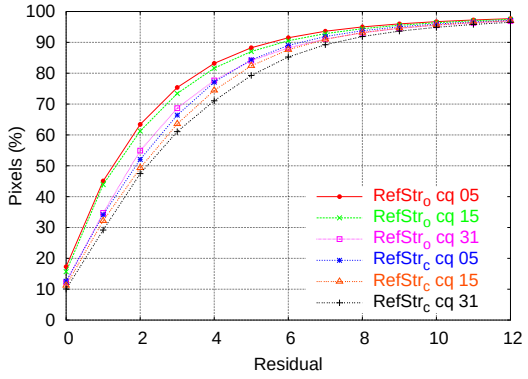
(a) Color Quantizer 5



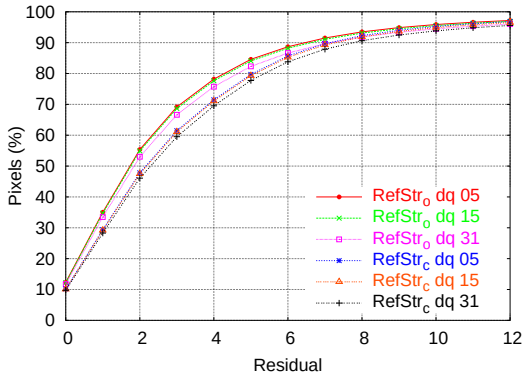
(d) Depth Quantizer 5



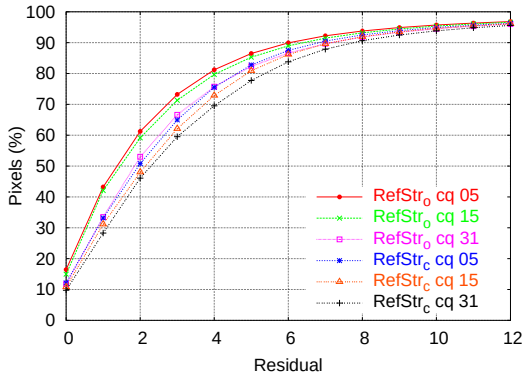
(b) Color Quantizer 15



(e) Depth Quantizer 15

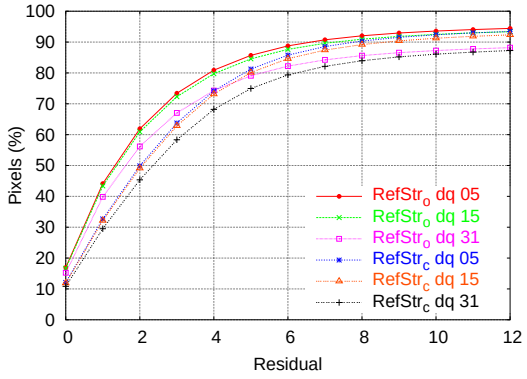


(c) Color Quantizer 31

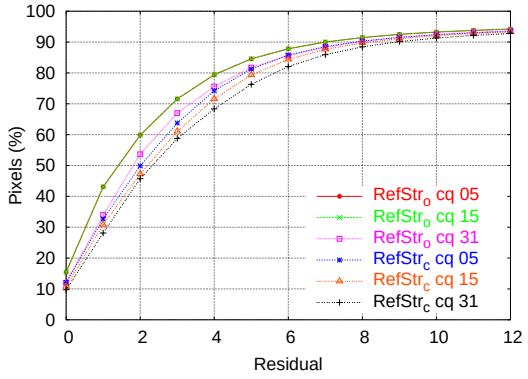


(f) Depth Quantizer 31

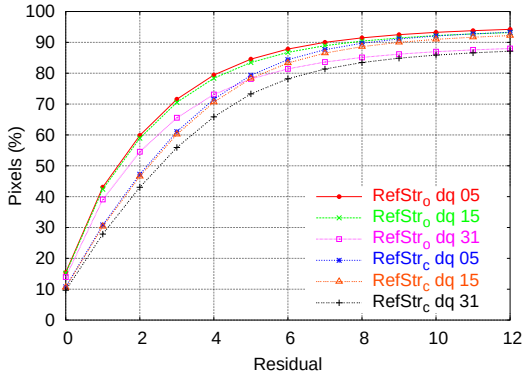
Figure 5.11: Breakdancers Accumulative Residual Histogram for *All Pixels*



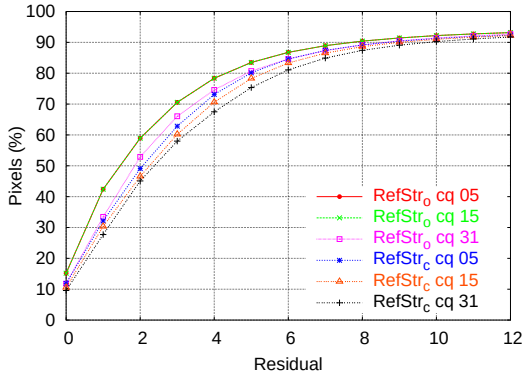
(a) Color Quantizer 5



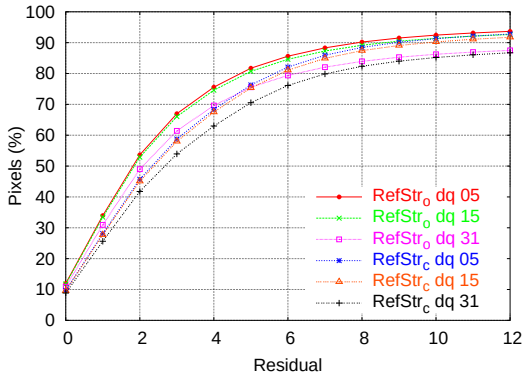
(d) Depth Quantizer 5



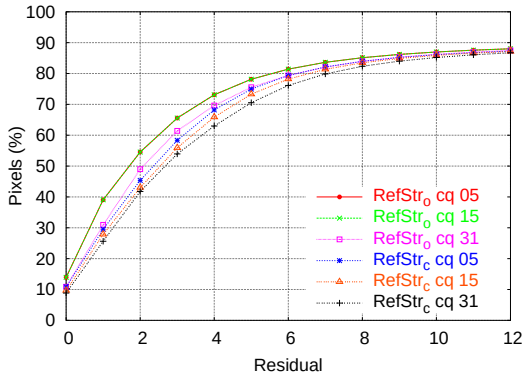
(b) Color Quantizer 15



(e) Depth Quantizer 15

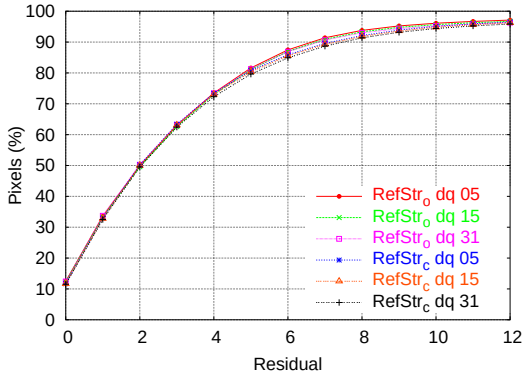


(c) Color Quantizer 31

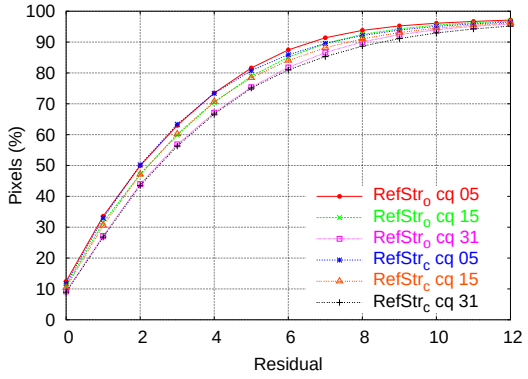


(f) Depth Quantizer 31

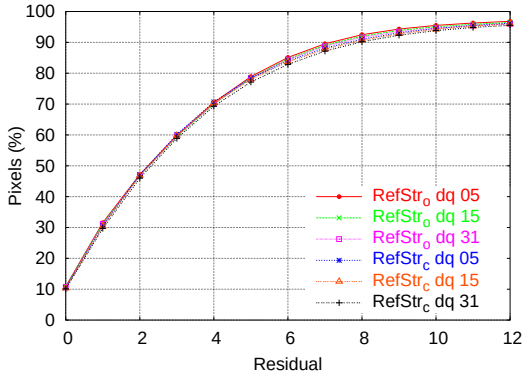
Figure 5.12: Breakdancers Accumulative Residual Histogram for *Spatial Reference Pixels*



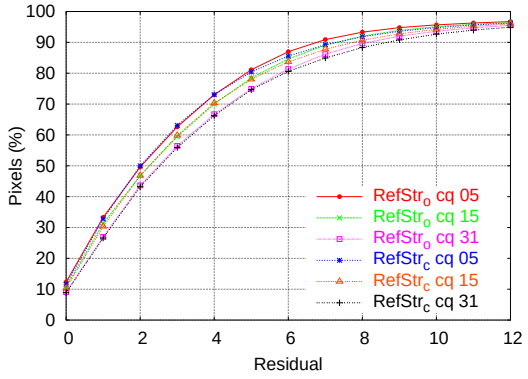
(a) Color Quantizer 5



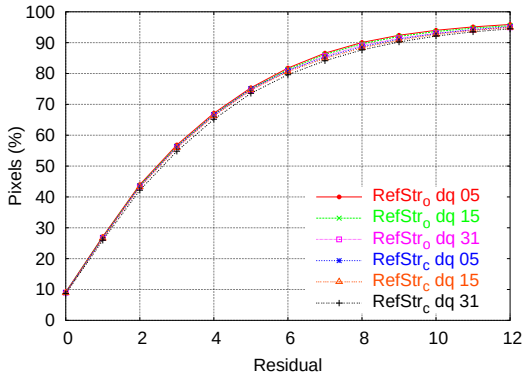
(d) Depth Quantizer 5



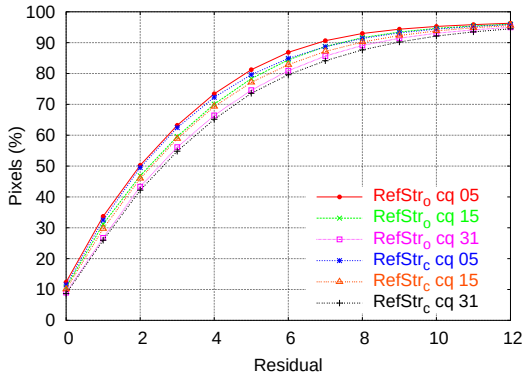
(b) Color Quantizer 15



(e) Depth Quantizer 15

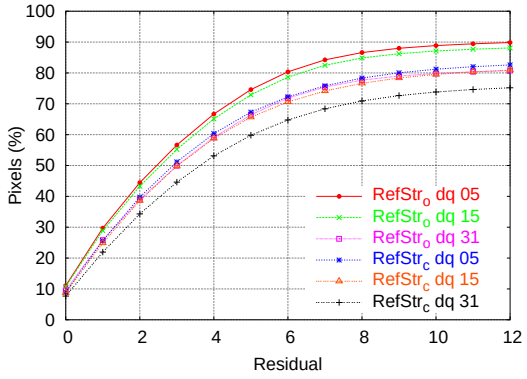


(c) Color Quantizer 31

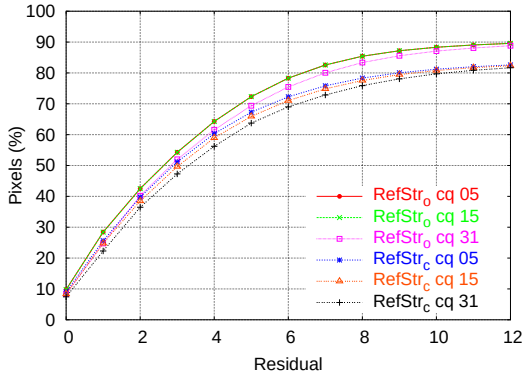


(f) Depth Quantizer 31

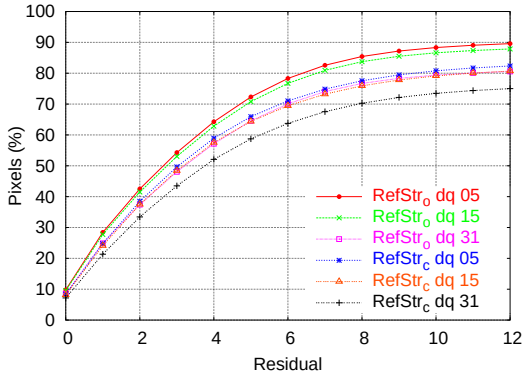
Figure 5.13: Ballet Accumulative Residual Histogram for *All Pixels*



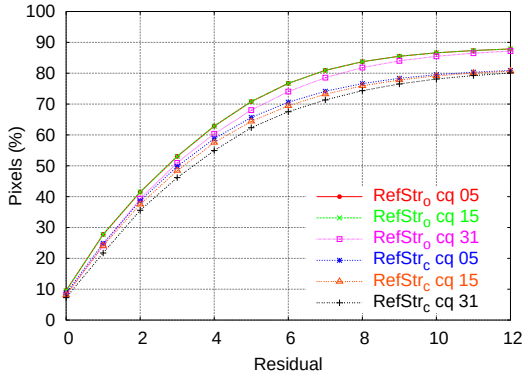
(a) Color Quantizer 5



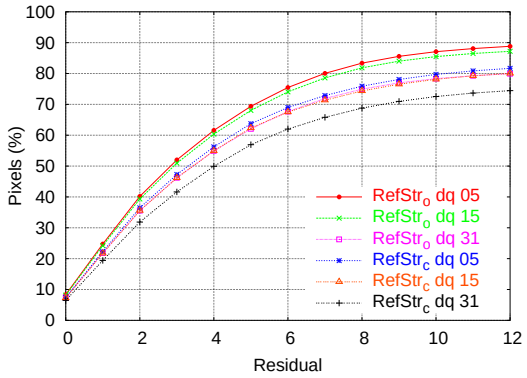
(d) Depth Quantizer 5



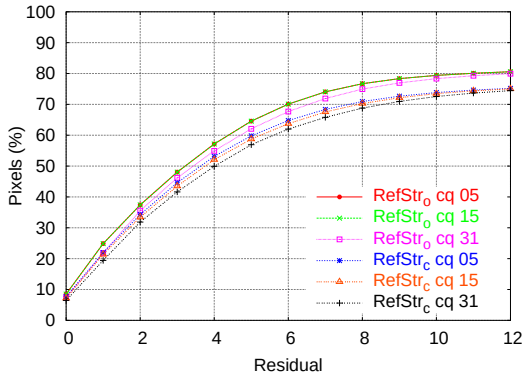
(b) Color Quantizer 15



(e) Depth Quantizer 15



(c) Color Quantizer 31



(f) Depth Quantizer 31

Figure 5.14: Ballet Accumulative Residual Histogram for *Spatial Reference Pixels*

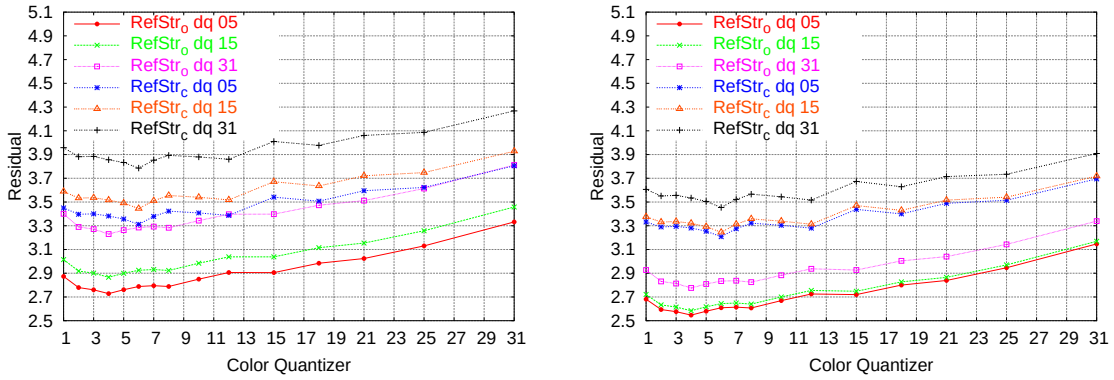
Figure 5.14 shows the average accumulative residual histogram of Ballet for *Spatial Reference Pixels*. Figures 5.14a-c shows the accumulative residual histogram for color quantizer 5, 15, and 31. Each figure has six different plots: $RefStr_o$ (streams 2 and 5) with depth quantizer 5 (dq 05), 15 (dq 15), and 31 (dq 31); and $RefStr_c$ (streams 0 and 7) with depth quantizer 5 (dq 05), 15 (dq 15), and 31 (dq 31). Figure 5.14d-f shows accumulative residual histogram for depth quantizer 5, 15, and 31 with six plots each : $RefStr_o$ (streams 2 and 5) with color quantizer 5 (cq 05), 15 (cq 15), and 31 (cq 31); and $RefStr_c$ (streams 0 and 7) with color quantizer 5 (cq 05), 15 (cq 15), and 31 (cq 31). As with all previous instances, $RefStr_o$ performs better than $RefStr_c$. However, compared to Breakdancers (Figure 5.12), the difference between $RefStr_o$ and $RefStr_c$ is larger for *Spatial Reference Pixels*, but for *All Pixels*, the difference between $RefStr_o$ and $RefStr_c$ is smaller. This is the result of Ballet having better temporally predicted values for occluded and peripheral pixels since it has slower motion than Breakdancers.

Residual Average and Standard Deviation

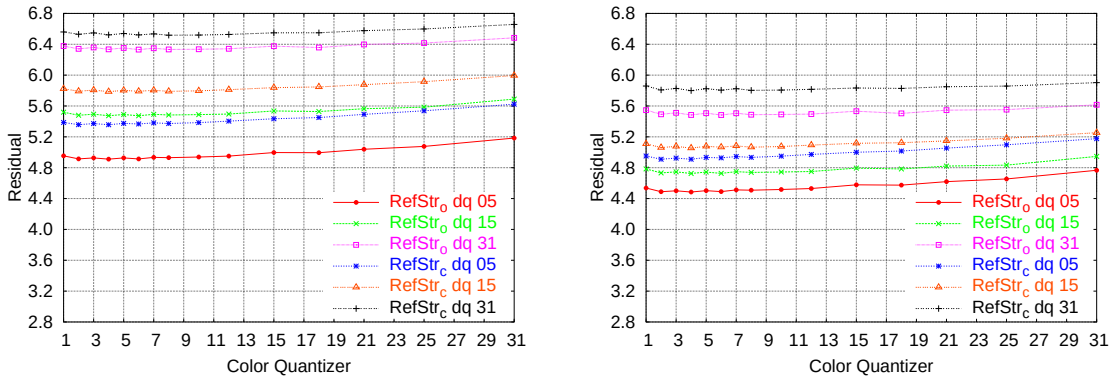
Figure 5.15 shows the average and standard deviation of the residuals for Breakdancers. The color quantizer is shown in the X-axis, and the residuals on the Y-axis. Each figure has six plots – $RefStr_o$ (streams 2 and 6) with depth quantizer 5 (dq 05), 15 (dq 15), and 31 (dq 31); and $RefStr_c$ (streams 0 and 7) with depth quantizer 5 (dq 05), 15 (dq 15), and 31 (dq 31). Figure 5.15a is the average for *All Pixels*, and Figure 5.15b the average for *Spatial Reference Pixels*. Figure 5.15c is the standard deviation for *All Pixels*, and Figure 5.15d is the standard deviation for *Spatial Reference Pixels*. Similarly, Figure 5.16 shows the average and standard deviation of the residuals for Ballet with six plots each: $RefStr_o$ (streams 2 and 5) with depth quantizer 5 (dq 05), 15 (dq 15), and 31 (dq 31); and $RefStr_c$ (streams 0 and 7) with depth quantizer 5 (dq 05),

Since temporal prediction for occluded and peripheral pixels is not always effective (Figure 5.9), for both data, the predicted images without occluded and peripheral pixels (*Spatial Reference Pixels*) have smaller residual average and standard deviation than the predicted images with occluded and peripheral pixels (*All Pixels*).

As with accumulative residual histograms, the predicted image from $RefStr_o$ has a smaller



(a) Average for *All Pixels* **(b) Average for *Spatial Reference Pixels***



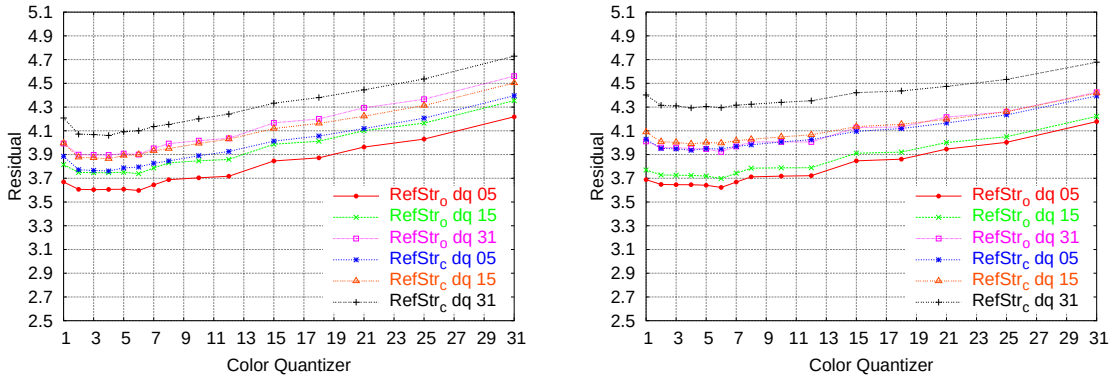
(c) Standard Deviation for *All Pixels* **(d) Standard Deviation for *Spatial Reference Pixels***

Figure 5.15: Residual Average and Standard Deviation for Breakdancers

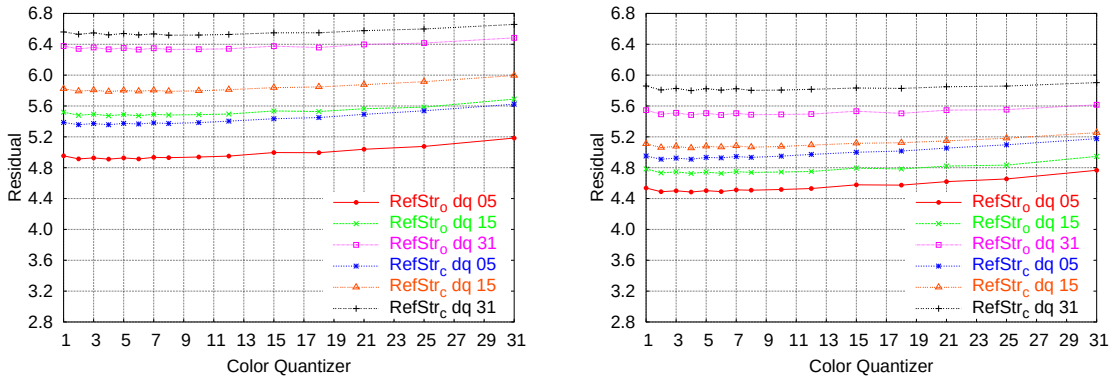
residual average and standard deviation than the predicted image from $RefStr_c$. This holds true for predicted images with and without occluded and peripheral pixels.

PSNR

The image quality of the predicted image is assessed using PSNR with the original frame as the base image. Figure 5.17a shows the PSNR for Breakdancers predicted image with occluded and peripheral pixels (*All Pixels*). The X-axis represents the color quantizer and the Y-axis represents PSNR. A plot is shown for $RefStr_o$ (streams 2 and 6) with depth quantizer 5 (dq 05), 15 (dq 15), and 31 (dq 31); and $RefStr_c$ (streams 0 and 7) with depth quantizer 5 (dq 05), 15 (dq 15), and 31 (dq 31). Figure 5.17b shows the same plots for PSNR of the Breakdancers predicted image without occluded and peripheral pixels (*Spatial Reference Pixels*). Figure 5.17c shows the PSNR for Ballet predicted image with occluded



(a) Average for *All Pixels* **(b) Average for *Spatial Reference Pixels***



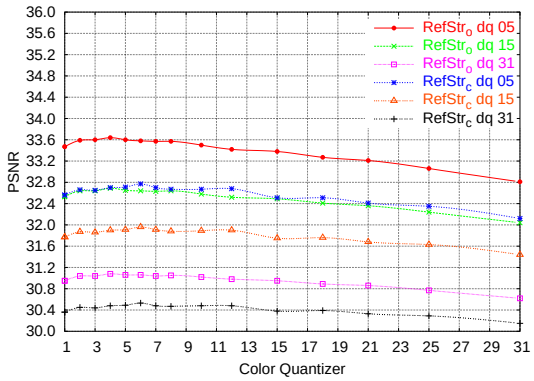
(c) Standard Deviation for *All Pixels* **(d) Standard Deviation for *Spatial Reference Pixels***

Figure 5.16: Residual Average and Standard Deviation for Ballet

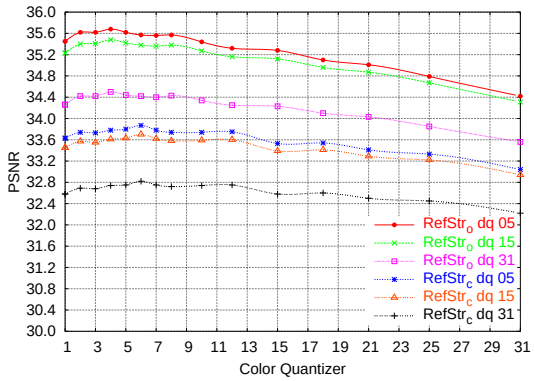
and peripheral pixels (*All Pixels*). A plot is shown for $RefStr_o$ (streams 2 and 5) with depth quantizer 5 (dq 05), 15 (dq 15), and 31 (dq 31); and $RefStr_c$ (streams 0 and 7) with depth quantizer 5 (dq 05), 15 (dq 15), and 31 (dq 31). Figure 5.17d shows the same plots for PSNR of the Ballet predicted image without occluded and peripheral pixels (*Spatial Reference Pixels*).

Comparable to residual average and standard deviation results, evaluating the predicted image without occluded and peripheral pixels result in a better PSNR than the predicted image with occluded and peripheral pixels, demonstrating that temporal prediction for occluded and peripheral pixels is not always effective (Figure 5.9). Furthermore, the predicted images from $RefStr_o$ have a better PSNR than predicted images from $RefStr_c$ for both with and without occluded and peripheral pixels.

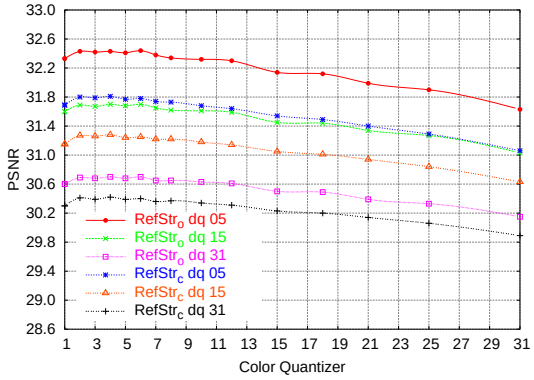
Comparing different metrics for predicted images – number of occluded and peripheral pixels, residual accumulative histogram, residual average and standard deviation, and PSNR



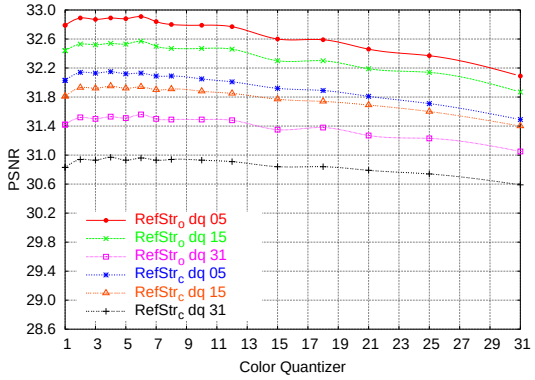
(a) Breakdancers for *All Pixels*



(b) Breakdancers for *Spatial Reference Pixels*



(c) Ballet for *All Pixels*



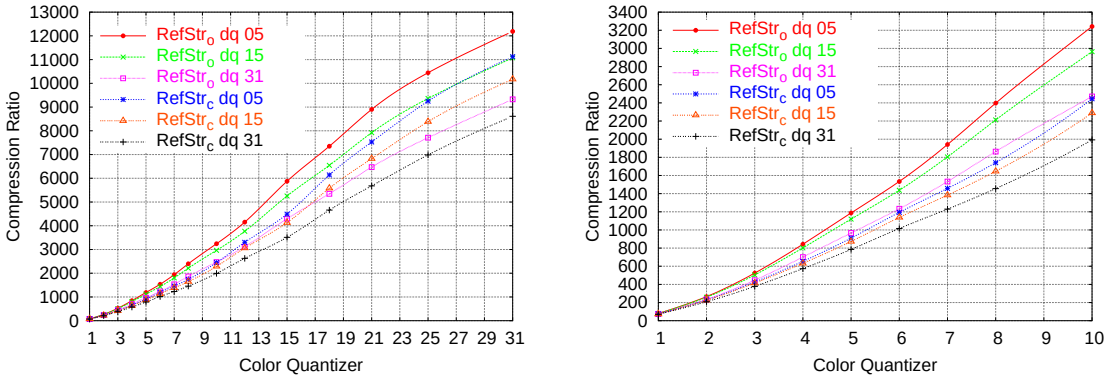
(d) Ballet for *Spatial Reference Pixels*

Figure 5.17: Predicted Image PSNR

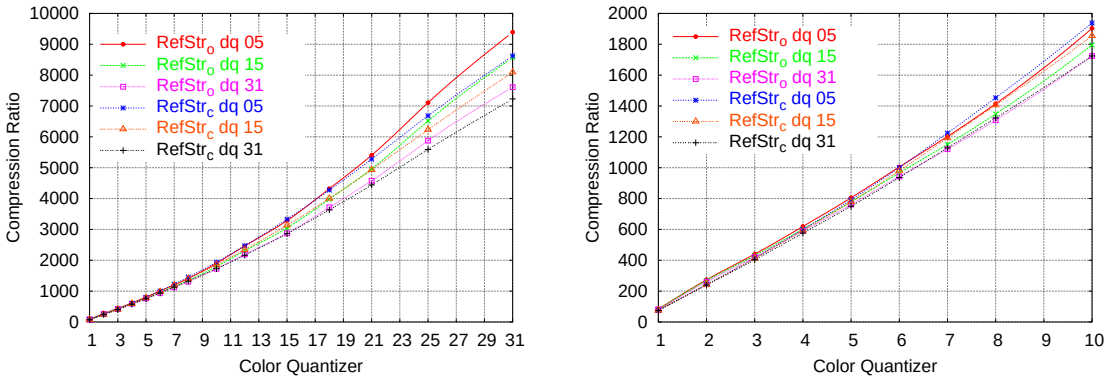
– shows that $RefStr_o$ performs better than $RefStr_c$. This is true for both Breakdancers and Ballet data, as well as for predicted images with and without occluded and peripheral pixels.

5.3.3 Residual Compression Ratio

In this section, the compression ratio of the resulting residuals is compared. The predicted image used for computing the residuals is generated from the two spatial reference frames and the temporal reference frame (Figure 5.8). The residuals are encoded using XviD. XviD is an open source ISO MPEG-4 [MPEG 2004] compliant video codec. XviD was selected mostly because of source code availability, which can be modified without any restrictions. The residuals are encoded with XviD as if the residuals had been computed using motion compensation. The resulting bitstream only contains the encoding for the residuals and no encoding for motion vectors.



(a) Breakdancers **(b) Closeup of Breakdancers**



(c) Ballet **(d) Closeup of Ballet**

Figure 5.18: Residual Compression Ratio

Figure 5.18 shows the average compression ratio for encoding residuals of the six non-reference streams. Figure 5.18a shows the compression ratio for the Breakdancers, with the X-axis representing the color quantizers and the Y-axis representing compression ratio. Six plots are shown in the figure – $RefStr_o$ (streams 2 and 6) with depth quantizer 5 (dq 05), 15 (dq 15), and 31 (dq 31); and $RefStr_c$ (streams 0 and 7) with depth quantizer 5 (dq 05), 15 (dq 15), and 31 (dq 31). Figure 5.18b is a closeup version of Figure 5.18a. Figure 5.18c shows the compression ratio for the Ballet. Six plots are shown in the figure – $RefStr_o$ (streams 2 and 5) with depth quantizer 5 (dq 05), 15 (dq 15), and 31 (dq 31); and $RefStr_c$ (streams 0 and 7) with depth quantizer 5 (dq 05), 15 (dq 15), and 31 (dq 31). Figure 5.18d is a closeup of Figure 5.18c.

Figure 5.18 demonstrates $RefStr_o$ has a higher compression ratio than $RefStr_c$. Furthermore, it demonstrates that the depth accuracy affects compression ratio – the more accurate the depth, the higher the compression ratio. This is to be expected, as the pixel depth be-

comes more accurate, the projection error of the pixel to the spatial reference frames becomes smaller.

5.4 Conclusion

The spatial coherence between streams greatly aids in effective encoding of multiple streams. In order to leverage this spatial coherence between streams, reference streams must be selected. Previously, reasonable streams were pre-selected by the user to be used as reference streams. However, as the number of streams increase, the task becomes non-trivial. In this chapter, an algorithm for reference streams selection was introduced and evaluated. Particularly, the following have been presented:

- Classification of the two approaches used for reference stream selection. Specifically, 1) select reference streams to maximize volume coverage, and 2) select reference streams to maximize volume overlap between reference streams and non-reference streams.
- Analysis of bad spatial prediction when using depth. Depth can be used to determine the accuracy of the spatial prediction and to identify occluded pixels. However, this is affected by the accuracy of depth values.
- Algorithm for selecting k reference streams from n streams that can handle stream configurations of large n.
- Methodology for evaluating reference streams using the predicted image and residual compression ratio.
- Evaluation of the two reference stream selection approaches using the proposed methodology. The experiments show that selecting reference streams to maximize volume overlap between reference streams and non-reference streams is superior to selecting reference streams to maximize volume coverage.

Chapter 6

Multiple Depth Stream Encoding

Multiple depth streams exhibit strong spatial coherence between streams since they capture the same environment from different angles. Also, just like video, multiple depth streams have strong temporal coherence between frames within each stream. In this chapter, encoding multiple depth streams using these spatial and temporal coherence are examined. To encode multiple depth streams, first, reference streams are selected using the algorithm from Chapter 5. *Reference stream*s are streams that serve as the basis streams for spatial prediction when encoding non-reference streams. Then the selected reference streams are encoded as intra-streams (I-Streams) with the algorithm presented in Chapter 3. Finally, non-reference streams are encoded as inter-streams (P-Streams) using the algorithm from Chapter 4.

The data sets used for multiple depth stream encoding in this chapter are the Ballet and Breakdancers data set from [Zitnick et al. 2004]. Both data sets have 8 depth streams which are 100 frames long at 1024×768 resolution. The streams were captured at 15 frames/sec (fps), and each frame has 24 bit color (RGB) and 8 bit depth information for each pixel. The depth was computed using the technique described in [Zitnick et al. 2004]. Example frames from both data sets are shown in Figure 6.1.

In Section 6.1, reference stream selection for multiple depth stream encoding is examined. Reference stream encoding is presented in Section 6.2, and non-reference stream encoding in Section 6.3. In the last Section 6.4, the results of multiple depth stream are presented.

(a) **Breakdancers Color**

(b) **Breakdancers Depth**

(c) **Ballet Color**

(d) **Ballet Depth**

Figure 6.1: Frame from Breakdancers and Ballet

6.1 Reference Stream Selection

When encoding multiple depth streams, to leverage the spatial coherence between streams, the first step is to select the reference streams. The selected reference streams serve as the basis streams for spatial prediction. Once the reference streams have been selected and encoded, non-reference streams can be encoded efficiently by spatially predicting from the reference streams.

In Chapter 5, two approaches for selecting reference streams were presented. One was to select streams to maximize volume coverage of the scene. The other was to select streams to maximize volume overlap between the reference streams and the non-reference streams. Selecting reference streams to maximize volume coverage emphasizes reducing peripheral pixels – non-reference stream pixels that project outside the reference stream. While selecting reference streams to maximize overlap between reference streams and non-reference streams emphasizes on reducing occluded pixels – non-reference stream pixels that is occluded in the reference stream and is not visible. The results from Chapter 5 indicates that selecting refer-

ence streams to maximize volume overlap between reference streams and non-reference streams is superior to selecting reference streams to maximize volume coverage.

The steps for selecting reference streams to maximize volume overlap are:

1. Generate sets of initial reference streams.
   (a) Sort the given streams.
   (b) Group the streams in all possible *reasonable* ways.
   (c) Find all possible *reasonable* candidate reference streams.
   (d) Generate all possible combinations of the candidate reference streams as sets of initial reference streams.
2. For each set of initial reference streams:
   (a) Designate the initial reference streams as the reference stream of each group.
   (b) Assign all other streams to a group. The streams are assigned to the group where the absolute angle of the stream and the reference stream is the smallest.
   (c) Compute a new reference stream for each group. The stream with the smallest *local squared angle sum* (LSAS) in the group is designated as the new reference stream of the group. An LSAS of a stream is the sum of the squared angle between the stream and all other streams in its group.
   (d) Repeat steps 2b - 2c until the reference streams converge and do not change between iterations. The resulting reference streams are designated as one of the candidates.
3. The final reference streams are selected by comparing the *total squared angle sum* (TSAS) of all candidates, and choosing the reference streams with the smallest TSAS. The TSAS is sum of the squared angle between the non-reference stream and it's reference stream for all non-reference streams.

A more detailed description for generating a set of initial reference streams can be found in Section 5.2.3. It discusses about stream sorting, initial group partitions, selecting reference

	Reference Streams	TSAS	Initial Reference Stream Sets
Breakdancers	2, 6	145.840	28
Ballet	1, 5	187.655	11
	2, 5	181.250	17

Table 6.1: Reference Stream Selection for Breakdancers and Ballet

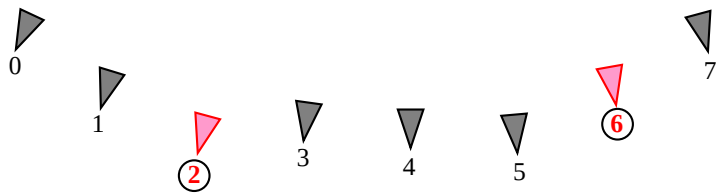
stream candidates, and generating a set of initial reference streams. Equations for calculating LSAS and TSAS, and a more detailed explanation for the metrics are presented in Section 5.2.1.

The results of reference stream selection for Breakdancers and Ballet data are shown in Table 6.1. The column *Reference Streams* shows the two selected reference streams. The column *TSAS* shows the TSAS value for the corresponding reference streams. The column *Initial Reference Stream Sets* shows the number initial reference stream sets that resulted in the specified reference streams from the reference streams selection algorithm. Two reference streams were selected because there were total of eight streams, and all 28 ($_8C_2$) possible combinations were used as set of initial reference streams.

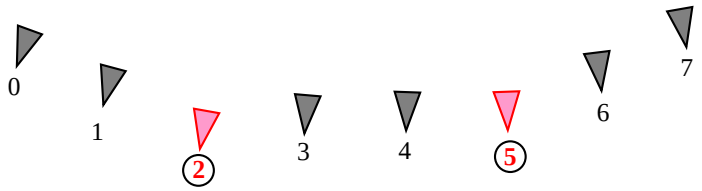
For Breakdancers data, all 28 possible initial reference stream sets resulted in one reference stream selection – streams 2 and 6. Figure 6.2a shows the placement of the streams for Breakdancers data with the selected reference streams shown in red and circled. For Ballet data, two possible reference stream selections were generated from 28 initial reference stream sets and the one with the smaller TSAS was selected – streams 2 and 5. This also had more initial starting points. Figure 6.2b shows the stream placement for the Ballet data with the reference streams in red and circled.

6.2 Reference Stream Encoding

Once the reference streams are selected, they need to be encoded. Since reference streams are referenced by non-reference streams for spatial prediction, reference streams need to be encoded such that it can be decoded independently of any other streams. Therefore reference streams are encoded as intra-streams (I-Streams). An intra-stream is encoded using the temporal coherence between frames within the same stream, but not the spatial coherence between streams. This enables intra-streams to be encoded and decoded individually from



(a) Breakdancers



(b) Ballet

Figure 6.2: Streams of Breakdancers and Ballet with Reference Streams Circled

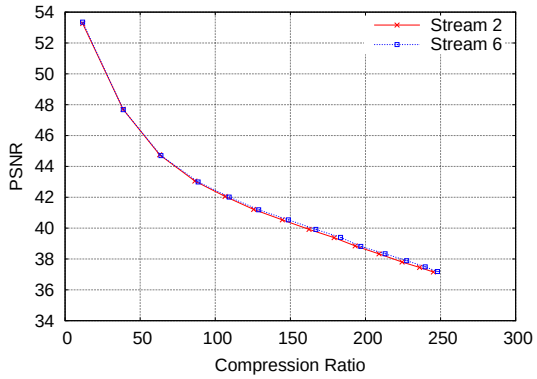
other streams.

Existing video encoding techniques utilize temporal coherence between frames by using motion compensation. Motion compensation incorporates temporal coherence between frames by predicting values from reference frames. *Motion vector*s are used to specify the predicted values from a reference frame. The difference between the actual value and the predicted value – *residual* – is encoded along with the motion vector. Superior motion vector result in smaller residuals, which increases encoding efficiency.

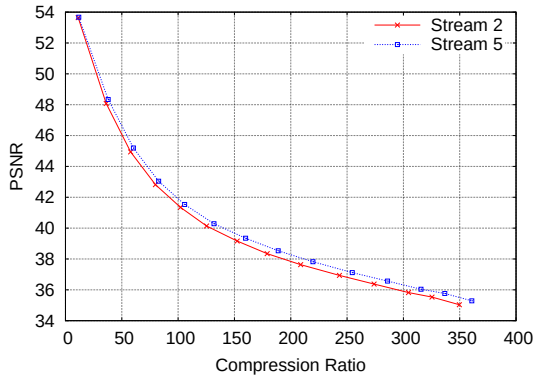
Motion compensation can be extended to encode a depth stream since it is very similar to traditional video – both streams have three channels of color to encode. The difference is that a depth stream has an extra channel – depth. In Chapter 3, three different schemes for extending the motion compensation for single stream video to encode intra-streams were investigated. They were:

- Use the motion vector based on color for encoding both depth and color,
- Use the motion vector based on depth for encoding both depth and color,
- Use separate motion vectors for encoding depth and color.

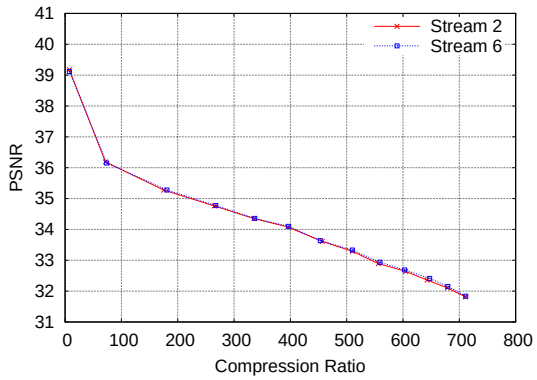
Results from Chapter 3 indicate that using separate motion vectors for encoding depth and color is generally superior to using a single motion vector. Figure 6.3 shows the results of



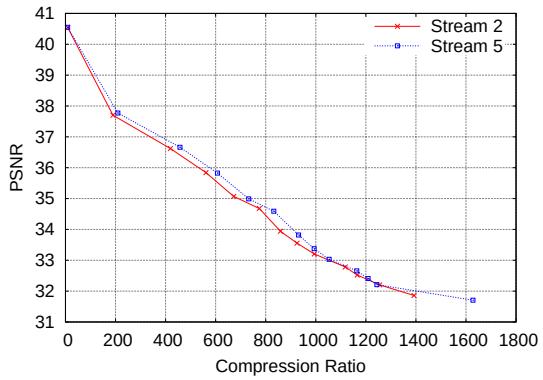
(a) Breakdancers Depth



(b) Ballet Depth



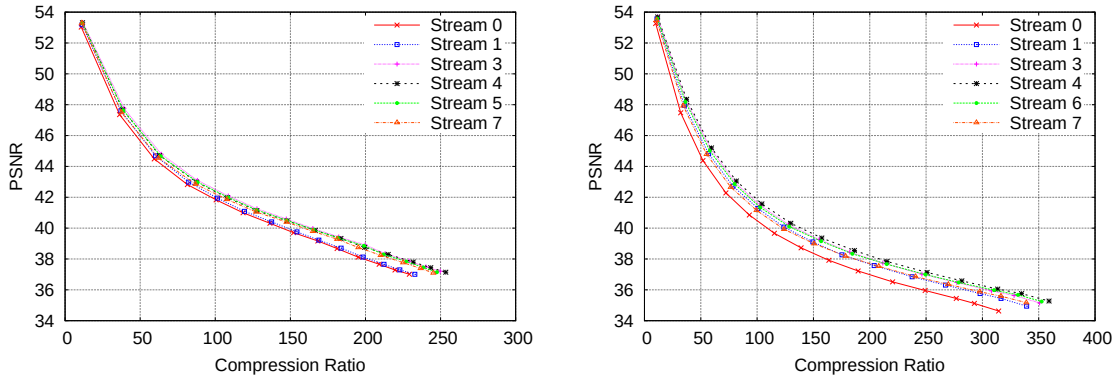
(c) Breakdancers Color



(d) Ballet Color

Figure 6.3: Reference Stream Encoding for Breakdancers and Ballet

encoding reference streams as I-Streams for Breakdancers and Ballet data – Streams 2 and 6 for Breakdancers, and Streams 2 and 5 for Ballet. The reference streams were encoded using separate motion vectors for color and depth. The reference streams were encoded using a modified version of a well known video codec XviD [XviD]. XviD is an open source ISO MPEG-4 [MPEG 2004] compliant video codec. The modified XviD codec is able to encode frames with four components – RGB and depth – using two motion vectors, one for RGB and one for depth. Only the first frame was encoded as an I-Frame. All other frames were encoded as P-Frames. For each figure, the X-axis represents the compression ratio, and the Y-axis shows the PSNR of the decoded image against the original image. Figure 6.3a shows the results for encoding of Breakdancers reference stream depth, Figure 6.3b for Ballet reference stream depth, Figure 6.3c for Breakdancers reference stream color, and Figure 6.3d for Ballet reference stream color. The Ballet data can be encoded at higher compression ratio than the Breakdancers to achieve the same quality. This is due to the fact that Breakdancers have



(a) Breakdancers Depth **(b) Ballet Depth**

Figure 6.4: Non-Reference Stream Depth Encoding for Breakdancers and Ballet

faster motion in the scene, and less temporal coherence between frames.

6.3 Non-Reference Stream Encoding

After encoding reference streams, the next step in encoding multiple depth streams is to encode the non-reference streams. As discussed in Chapter 4, non-reference streams are encoded as inter-streams (P-Streams). The fact that multiple depth streams capture the same environment from different angles and exhibit spatial coherence between streams is utilized in encoding inter-streams: The non-reference streams (inter-streams) predict reference values from reference streams, and encode the difference – *residual*.

When encoding a non-reference stream, the depth values associated with each pixel is used to project it to the frame of the reference stream. If the color of the corresponding pixels in the reference stream is similar to the color of the pixel in the non-reference stream, the residual and the depth is used to encode the pixel color of the non-reference stream. Therefore, for non-reference streams, depth is first encoded independently of color.

From Section 4.1, for inter-stream depth encoding, the results indicated that only using temporal coherence information was preferable than using only spatial coherence information, or using both temporal and spatial coherence information. Figure 6.4 shows the results for depth encoding of non-reference streams only using temporal coherence for Breakdancers and Ballet data. The Breakdancers non-reference streams – Streams 0, 1, 3, 4, 5, and 7 – is shown in Figure 6.4a, and the Ballet non-reference streams – Streams 0, 1, 3, 4, 6, and 7 –

in Figure 6.4b. The X-axis represents the compression ratio, and the Y-axis shows the PSNR of the decoded depth against the original depth. Depth was encoded using a modified XviD codec, identical to the one used in Section 4.1. The modified XviD codec is able to encode frames with only one component instead of three. Similar to reference streams in the previous section, only the first frame was encoded as an I-Frame, and all other frames were encoded as P-Frames.

Once non-reference stream depth has been encoded, color of non-reference stream is encoded by predicting a color for each pixel and encoding the residual between the predicted color and the actual color. The predicted color is derived by utilizing the following information:

- Depth values from the current frame to encode,
- Depth values from the temporal reference frame,
- Depth values from the spatial reference frames,
- Color values from the temporal reference frame,
- Color values from the spatial reference frames.

These information is used to create a *reference mask* for each frame of the non-reference stream. A reference mask indicates which reference frame, either spatial or temporal, should be used to predict the color value for the given pixel. A reference mask is encoded as a list of the following four primitives:

- Consensus filter,
- Delta depth threshold,
- List of boxes,
- List of pixels.

For a detailed explanation of the primitives, refer to Section 4.2.1. The list of primitives are processed in order to assign predicted colors for each pixel. Once a predicted color has

been assigned for a pixel, following primitives in the list do not change the predicted color for the pixel: Following primitives only assign predicted colors for pixels that is unknown.

The list of primitives for the reference mask is generated by the following algorithm.

1. Initialization.

   (a) Use the consensus filter.

   (b) Create a list of boxes where temporal reference color should be used as the predicted color.

   (c) For the remaining pixels, delta depth threshold primitives are used. Delta depth threshold primitives which minimizes error is repeatedly selected until all pixels have a predicted color. Selecting a delta depth threshold of 255 with the temporal reference frame will guarantee all pixels have a predicted color.

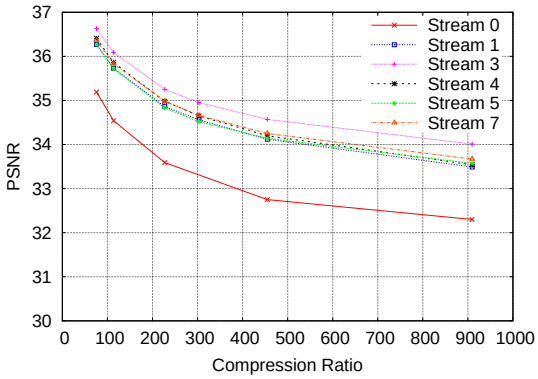
2. Error Correction.

   (a) Identifies pixels with largest error introduced in the initialization.

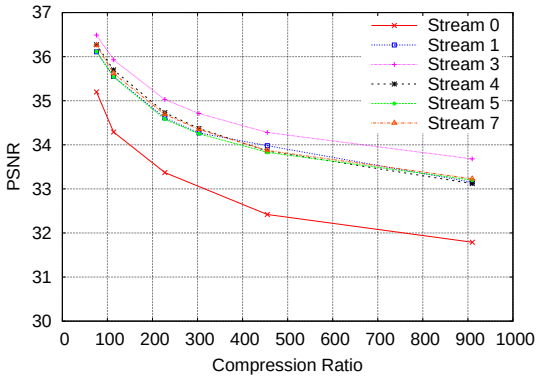
   (b) Create primitives – list of pixels or list of boxes – to correct the pixels with the largest error. Then insert these primitives in the list before the primitive that caused the error.

After encoding the reference mask, the residual – difference between the predicted value generated from the encoded reference mask and the actual value – is encoded. The color residuals are represented in YCbCr color space, one luminance and two chrominance channels, and the residual for each channel is encoded separately. The bit allocation for luminance and two chrominance channel is 4 to 1 since the human eye is more sensitive to luminance than chrominance. The residuals are encoded using JPEG2000 [JPEG 2000], a discrete wavelet transform based codec.

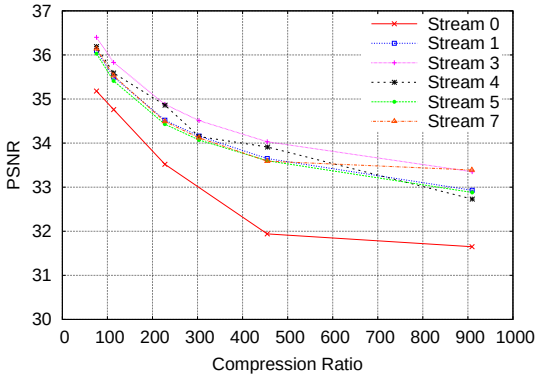
Figure 6.5, Figure 6.7, and Figure 6.9 show the results for color encoding of the Breakdancers non-reference streams – Streams 0, 1, 3, 4, 5, and 7. Figure 6.5 compares the Breakdancers non-reference stream color encoding with reference stream color quantizer set to 5 and depth quantizer for reference streams and non-reference streams set to various values –



(a) Depth Quantizer 5

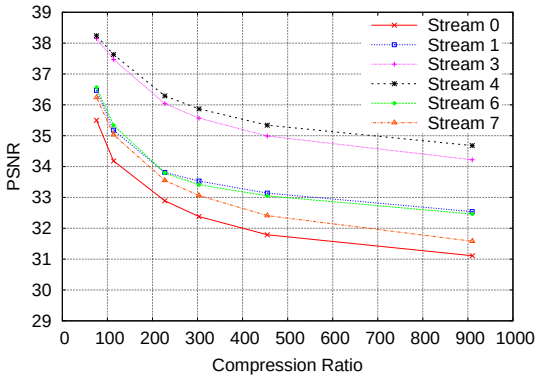


(b) Depth Quantizer 15

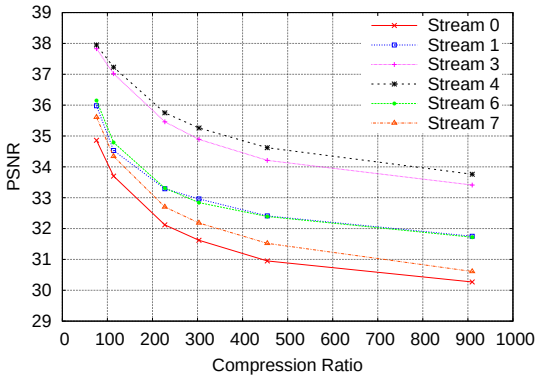


(c) Depth Quantizer 25

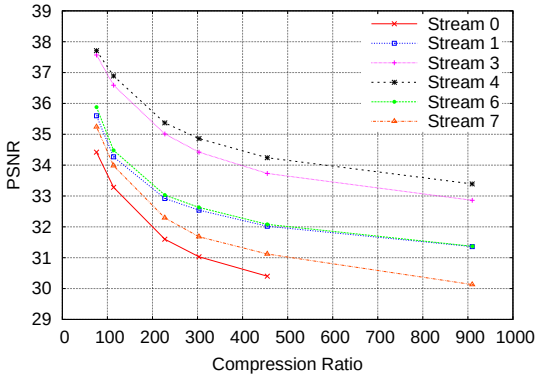
Figure 6.5: Breakdancers Reference Stream Color Quantizer 5



(a) Depth Quantizer 5

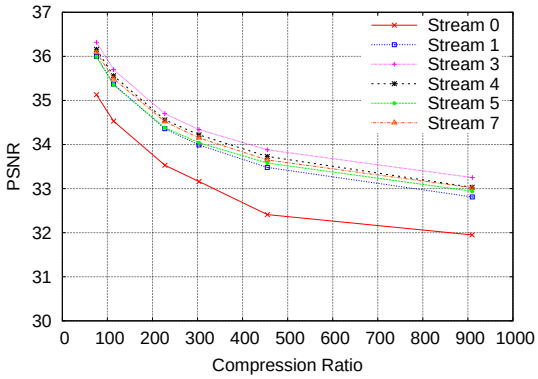


(b) Depth Quantizer 15

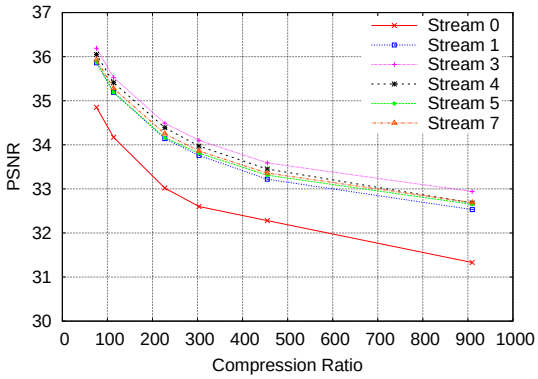


(c) Depth Quantizer 25

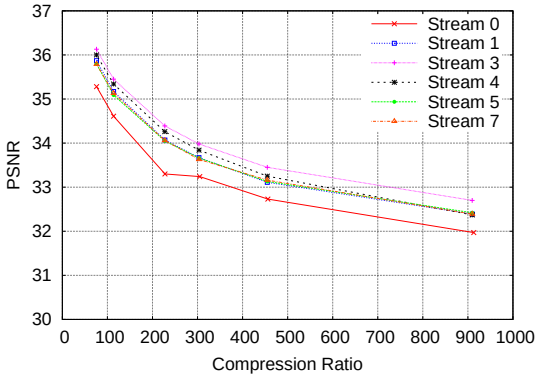
Figure 6.6: Ballet Reference Stream Color Quantizer 5



(a) Depth Quantizer 5

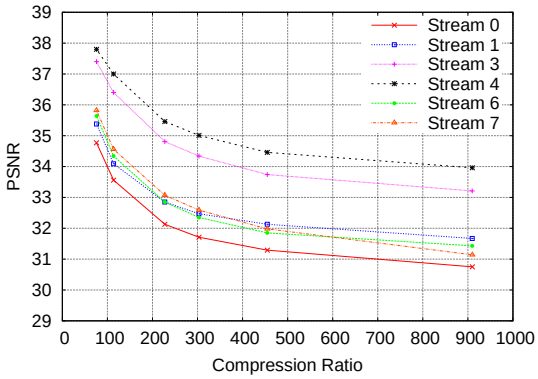


(b) Depth Quantizer 15

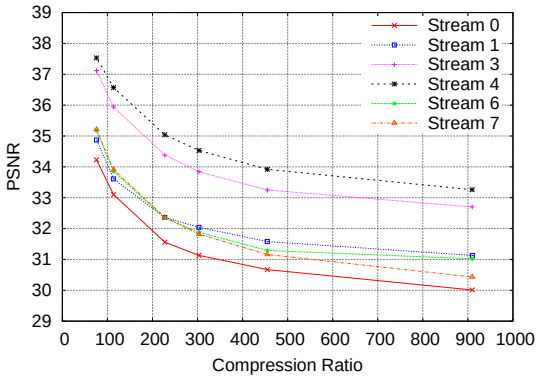


(c) Depth Quantizer 25

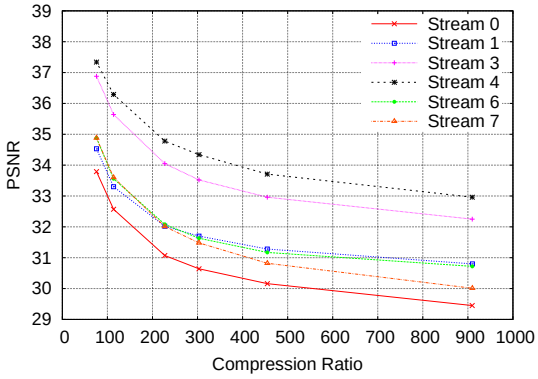
Figure 6.7: Breakdancers Reference Stream Color Quantizer 15



(a) Depth Quantizer 5

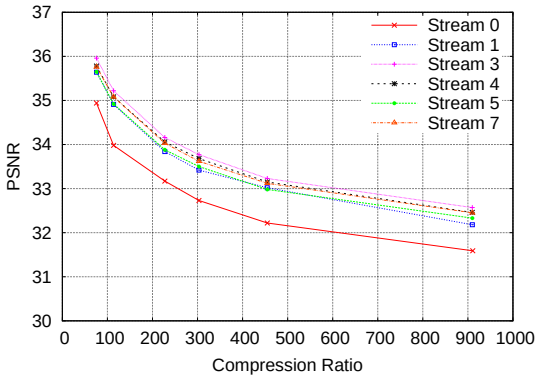


(b) Depth Quantizer 15

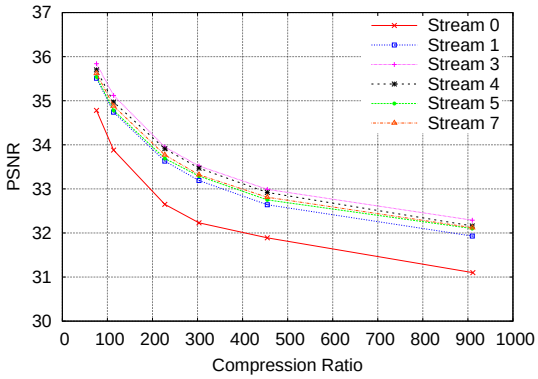


(c) Depth Quantizer 25

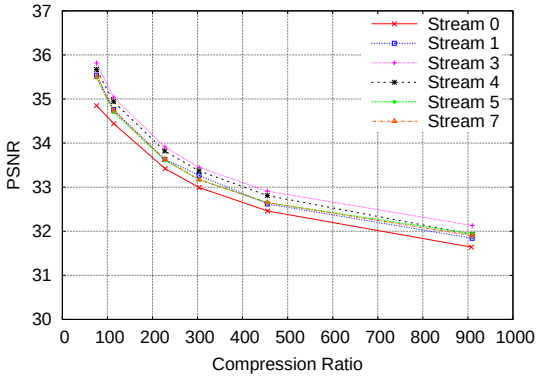
Figure 6.8: Ballet Reference Stream Color Quantizer 15



(a) Depth Quantizer 5

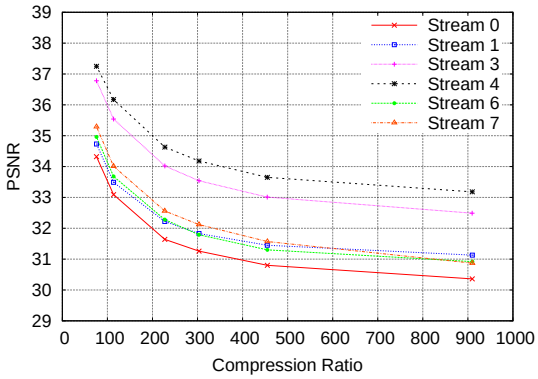


(b) Depth Quantizer 15

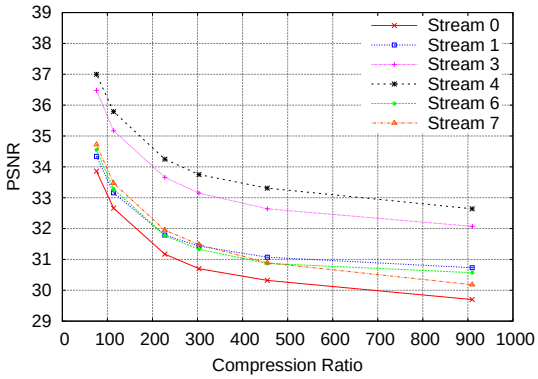


(c) Depth Quantizer 25

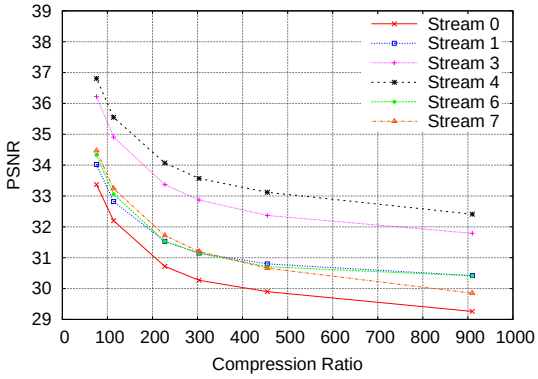
Figure 6.9: Breakdancers Reference Stream Color Quantizer 25



(a) Depth Quantizer 5



(b) Depth Quantizer 15



(c) Depth Quantizer 25

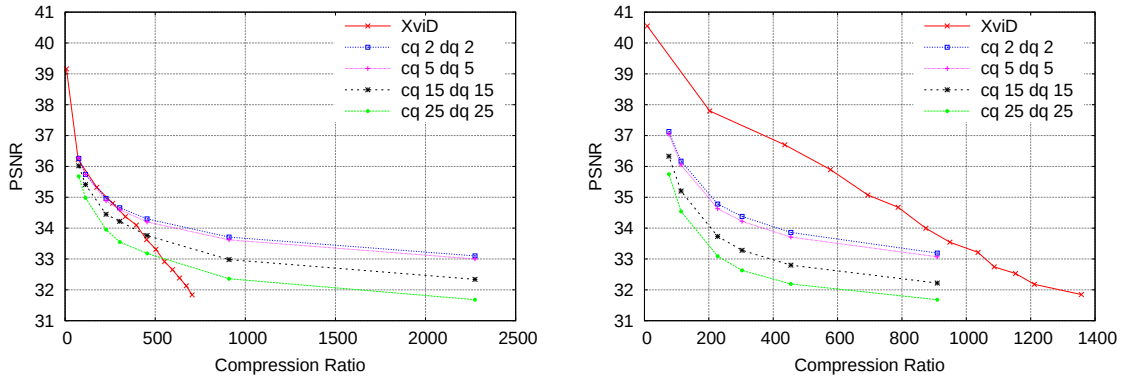
Figure 6.10: Ballet Reference Stream Color Quantizer 25

Figure 6.5a at 5, Figure 6.5b at 15, and Figure 6.5c at 25. Similarly, Figure 6.7 compares the Breakdancers non-reference stream color encoding with reference stream color quantizer set to 15 and depth quantizer for reference streams and non-reference streams set to different values – Figure 6.7a at 5, Figure 6.7b at 15, and Figure 6.7c at 25, – and Figure 6.9 compares the Breakdancers non-reference stream color encoding with reference stream color quantizer set to 25 and depth quantizer for reference streams and non-reference streams set to various values – Figure 6.9a at 5, Figure 6.9b at 15, and Figure 6.9c at 25.

The results for color encoding of the Ballet non-reference streams – Streams 0, 1, 3, 4, 6, and 7 – are shown in Figure 6.6, Figure 6.8, and Figure 6.10. Figure 6.6 compares the Ballet non-reference stream color encoding with reference stream color quantizer set to 5 and depth quantizer for reference streams and non-reference streams set to various values – Figure 6.6a at 5, Figure 6.6b at 15, and Figure 6.6c at 25. Figure 6.8 compares the Ballet non-reference stream color encoding with reference stream color quantizer set to 15 and depth quantizer for reference streams and non-reference streams set to different values – Figure 6.8a at 5, Figure 6.8b at 15, and Figure 6.8c at 25, – and Figure 6.10 compares the Ballet non-reference stream color encoding with reference stream color quantizer set to 25 and depth quantizer for reference streams and non-reference streams set to various values – Figure 6.10a at 5, Figure 6.10b at 15, and Figure 6.10c at 25.

For all figures in Figure 6.5, Figure 6.6, Figure 6.7, Figure 6.8, Figure 6.9, and Figure 6.10, the X-axis represents the compression ratio, and the Y-axis shows the PSNR of the decoded color frames against the original color

For Breakdancers, stream 0 results were inferior to other non-reference streams since it was the furthest from the reference streams. Streams 1 and 3 were closest to reference stream 2, streams 5 and 7 were closest to reference stream 6, and stream 4 was between reference streams 2 and 6. For Ballet, streams 3 and 4 had superior results than other non-reference streams since it was the closest to the reference streams – streams 2 and 5. Streams 1 and 6 generally resulted in higher quality encoding for same compression ratio since both were closer to the reference streams than streams 0 and 7, but as the depth and color quality of the reference streams decreased the difference also decreased.



(a) Breakdancers **(b) Ballet**

Figure 6.11: Non-Reference Stream Color Encoding

6.4 Results

In this section, the results of encoding multiple depth stream are presented. First, the results of encoding multiple depth stream using temporal and spatial encoding is compared to encoding multiple depth stream using only temporal coherence. Then the results of encoding multiple depth stream at different compression ratios are given.

Since reference streams and non-reference stream depth are encoded only using temporal coherence, only the non-reference stream color encoding is compared between using temporal coherence, and using spatial and temporal coherence. For using temporal coherence, the XviD codec was used to encode the non-reference stream color. Figure 6.11 compares the results for encoding non-reference stream color using XviD and using inter-stream encoding algorithm. Inter-Streams were encoded at various compression ratios with different qualities of reference streams: reference stream color quantizer 2 and depth quantizer 2 for reference streams and non-reference streams (cq 2 dq 2), color quantizer 5 and depth quantizer 5 (cq 5 dq 5), color quantizer 15 and depth quantizer 15 (cq 15 dq 15), and color quantizer 25 and depth quantizer 25 (cq 25 dq 25).

For both Breakdancers and Ballet data, non-reference streams (Figure 6.11) were encoded as P-Streams using two reference streams – streams 2 and 6 for Breakdancers, and streams 2 and 5 for Ballet. The reference streams were encoded as I-Streams using the same modified XviD codec presented in Section 6.2, and only the first frame was encoded as an I-Frame while all other frames were encoded as P-Frames. Depth for the non-reference streams

were encoded with the modified XviD used in Section 6.3. As with I-Stream encoding, only the first frame was encoded as an I-Frame and all other frames were encoded as P-Frames. For each figure, the X-axis represents the compression ratio, and the Y-axis shows the PSNR of the decoded image against the original image.

For Breakdancers (Figure 6.11a), only when the reference streams and depth are encoded with high quantizers (cq 15 dq 15 and cq 25 dq 25) and P-Stream is encoded at low compression ratio, does XviD encoding have a higher PSNR than P-Stream encoding. This case would be highly unlikely to occur, since it means that the P-Streams would be encoded at a higher quality than the reference streams. So for Breakdancers, P-Stream encoding is superior to XviD. However for Ballet, using XviD encoding significantly outperforms P-Stream encoding. This is the effect of Breakdancers having more and faster motion changes in the scene than Ballet. In fact, when encoding Ballet with XviD, more than 90% of the blocks were encoded using motion compensation while for Breakdancers, less than 80% were encoded using motion compensation [Kum and Mayer-Patel 2006]. So for Ballet, there was not much more to be gained from spatial prediction.

Table 6.2 shows the results of multiple depth stream encoding for Breakdancers and Ballet at different compression ratios. *Reference Stream* column shows the average frame length in bytes/fr (Fr. Length), compression ratio (Comp. Ratio), the PSNR for color (Color PSNR), and PSNR for depth (Depth PSNR). *Non-Reference Stream Depth* column shows the average frame length in bytes/fr (Fr. Length), compression ratio (Comp. Ratio), and PSNR. The *Non-Reference Stream Color* column shows the average frame length in bytes/fr (Fr. Length), the average reference mask length in bits/fr (Mask), compression ratio (Comp. Ratio), and PSNR. The last column *Total Comp. Ratio* shows the compression ratio for encoding for all eight depth streams.

	Reference Stream				Non-Reference Stream Depth			Non-Reference Stream Color				Total Comp. Ratio
	Fr. Length (bytes/fr)	Comp. Ratio	Color PSNR	Depth PSNR	Fr. Length (bytes/fr)	Comp. Ratio	PSNR	Residual (bytes/fr)	Mask (bits/fr)	Comp. Ratio	PSNR	
Breakdancers	97597.23	31.07	37.16	49.59	29923.89	25.33	49.50	19998.20	961.70	113.71	35.74	49.03
	80311.83	37.75	37.16	44.73	12211.82	62.07	44.64	9999.70	1037.26	227.41	34.88	82.54
	24812.21	122.20	35.34	44.73	12211.82	62.07	44.64	7498.80	1042.82	303.25	34.51	144.48
	10856.90	279.28	34.21	39.92	4702.02	161.21	39.83	4999.40	1231.33	454.86	33.47	303.50
	7713.81	393.07	33.31	37.47	3250.12	233.23	37.39	2498.20	1250.30	910.27	32.37	485.93
Ballet	59330.95	51.10	38.42	49.99	31383.81	24.15	49.81	19999.00	810.58	113.71	36.17	101.58
	42157.69	71.92	38.42	45.07	13383.92	56.64	44.89	9998.60	883.61	227.44	34.52	167.69
	42157.69	71.92	38.42	45.07	13383.92	56.64	44.89	7499.40	890.94	303.23	34.09	187.09
	18034.81	168.12	36.64	45.07	13383.92	56.64	44.89	4999.60	910.00	454.85	33.45	365.27
	7138.68	424.74	34.76	38.44	4255.63	178.12	38.28	2499.70	1168.25	909.73	31.68	799.38

Table 6.2: Multiple Depth Stream Encoding

Chapter 7

Conclusions

In this dissertation, I have examined and presented work on encoding multiple depth streams using spatial coherence between streams and temporal coherence between frames within a stream. A *depth stream* is a stream of frames which has color and depth information per pixel that is used for representing dynamic environments. I conclude the dissertation by presenting a summary of the research and a discussion of future works in this chapter.

7.1 Summary

When using multiple depth streams to represent dynamic environments, uncompressed multiple depth streams require significant storage space and transmission bandwidth – 8 depth streams with 1024×768 resolution and 32 bits per pixel (1 byte each for RGB and depth) at 15 frames per second requires a bandwidth of 2.8 Gb/sec. Therefore, multiple depth streams need to be compressed to reduce the required disk storage and network bandwidth for streaming. Since multiple depth streams capture a dynamic environment from different angles, they exhibit spatial and temporal coherence that can be utilized for effective encoding.

The first step in utilizing spatial coherence between streams for encoding multiple depth streams is to select *reference stream*s. Reference streams are basis streams used for spatial prediction by non-reference streams.

In Chapter 5, I have classified two approaches for selecting reference streams: One is to maximize reference stream volume coverage of the environment, and the other is to maximize volume overlap between reference streams and non-reference streams. I have also introduced

methodology using the predicted image (image of predicted reference values used for encoding) and residual compression ratio for evaluating the selected reference streams. Using these methodologies, I have evaluated the two different approaches for selecting reference streams which show that selecting reference streams to maximize volume overlap between reference streams and non-reference streams is superior to maximizing volume coverage of reference streams. In addition, I have presented an algorithm for selecting k reference streams from n streams to maximize volume overlap between reference streams and non-reference streams. Since the exact solution is very expensive when n is large, the presented algorithm generates an approximate solution efficiently, even for large n.

Once reference streams are selected, the next step in encoding multiple depth streams is to encode the reference streams. Since reference streams serve as the basis for non-reference streams, the reference streams are encode independently: They are encoded as *intra-stream*s (*I-Stream*s). I-Streams are encoded using only the temporal coherence between frames within the stream so they can be encoded and decoded independent of other streams. Additionally, depth streams are very similar to traditional videos in that both contain streams of color frames. Therefore, I-Streams can be encoded by extending traditional video encoding techniques to also encode additional depth information.

In Chapter 3, I have investigated methods for extending traditional video encoding techniques to encode I-Streams. Specifically, I have examined three different methods to extend motion compensation used for video encoding to encode I-Streams. One is to only use the motion vector based on color to encode color and depth. Another is to only use the motion vector based on depth to encode color and depth. The last is to use separate motion vectors to encode color and depth. I have also introduced a methodology to evaluate the three different methods – the *accumulative residual histogram*. *Residual* is the difference between the predicted value from motion compensation and the actual value, and an accumulative residual histogram shows the percent of pixels in the image that have equal or less than the given residual. From the evaluation of the three different approaches, encoding I-Streams using separate motion vectors for color and depth generally yielded superior results than the other two methods. Even though extra bits were needed to encode an extra motion vector, using separate motion vectors generated more suitable reference values which reduced the bits needed to

encode the residual.

The final step in encoding multiple depth streams is to encode the non-reference streams. Since non-reference streams can utilize reference streams for spatial prediction, they are encoded as *inter-stream*s (*P-Stream*s). P-Streams are encoded using the spatial reference frames from the reference streams as well as the temporal reference frame within the stream. Since depth streams have depth values per pixel, the depth is used to predict a reference value on an individual pixel basis instead of using motion compensation which predicts on a block basis. Using depth, a pixel can be projected into the spatial reference frame to find the corresponding pixel which can be used to predict a reference value. Therefore, P-Stream depth values should be encoded independent of color as they are essential for color prediction.

In Chapter 4, I have investigated methods for encoding P-Stream depth values independent of color. The investigation showed that, for encoding P-Stream depth, only using temporal coherence between frames was preferable to only using spatial coherence between streams or using both spatial and temporal coherence. I have also introduced the *reference mask* (mask that indicates, per pixel, which reference frame to use for prediction) to effectively encode P-Stream color using spatial and temporal prediction. Utilizing the color and depth values from spatial reference frames and temporal reference frame, as well as the depth value for the encoding frame, the reference mask can be encoded effectively using four primitives that I have proposed – *consensus filter*, *delta depth threshold*, *list of boxes*, and *list of pixels*. Finally, I have evaluated the P-Stream color encoding against traditional video encoding. It showed that for environment with relatively fast motion, P-Stream encoding is preferable. But for environment with relatively slow motion, encoding the streams using traditional video encoding performed much better than P-Stream encoding. The overhead of spatial prediction was significant compared to the benefits.

7.2 Future Work

This dissertation presents the initial step in encoding multiple depth streams for real-time encoding and streaming of dynamic environments. In this section, I present additional improvements that can be made to the encoding algorithm along with potential areas for future

work.

7.2.1 Encoding Improvements

In this section, I discuss potential methods for improving multiple depth stream encoding.

Global Background

Incorporating additional reference frames to the proposed multiple depth stream encoding is trivial. The additional cost of encoding would be the increase in reference mask bits to specify the reference frame. One additional reference frame that would be interesting to add would be the *global background*.

If the cameras acquiring the scene is stationary, just like Breakdancers and Ballet data from [Zitnick et al. 2004], adding a global background to use as a reference frame should improve the encoding efficiency. Also generating a global background should not be too difficult since most systems can initially acquire a background scene before starting to acquire the actual dynamic environment. It would be interesting to compare the encoding overhead of the global background frame against the gains in using a global background frame for encoding.

Depth

When encoding depth, either for I-Streams or P-Streams, the color quantizer matrix used for color was also used for depth. Ordinarily, the color quantizer matrix is based on human perception [MPEG 2004]. However, depth values do not have the same perceptual characteristics as color. Experiments to define an optimal quantizer matrix for depth should improve encoding efficiency.

Furthermore, when encoding P-Stream depth, instead of utilizing only temporal coherence, a hybrid approach can be used. Since P-Stream depth is encoded on a block basis, and has spatial and temporal reference frames, each block can be encoded independently. Each block can be encoded either with temporal prediction or encoded with spatial prediction. Using this hybrid approach, an overhead to specify the block type will be needed but it should improve encoding due to better predicted reference values.

(a) Encoding Frame

(b) Predicted Image

Figure 7.1: Encoding Frame and Predicted Image

I-Stream

When encoding I-Streams, it was shown in Section 3.2 that using separate motion vectors for color and depth did not always yield superior results than using a single motion vector. Therefore, it should be more efficient to indicate if one or two motion vectors are present on a block basis. The overhead of specifying the number of motion vectors per block should easily be overcome if enough blocks can be encoded with just one motion vector.

P-Stream Motion Compensation

Using temporal prediction without motion compensation and spatial prediction to encode P-Streams do not always generate suitable reference values – e.g. the artificial arm of the ballerina between the actual arms in Figure 7.1b. Investigating methods to incorporate more advanced temporal prediction techniques, such as motion compensation, should improve encoding.

One method is to introduce a new primitive – *motion compensated box* – for encoding reference masks. A motion compensated box is a box with a motion vector to indicate the location of the predicted values in the reference frame. If motion compensated boxes are only used for error correction step in the reference mask encoding and only for regions with large residuals, it should minimize the number of motion vector searches performed.

Another interesting method to consider is to use the predicted image (Figure 7.1b) generated from the reference mask as the reference frame for motion compensation. Since the predicted image is from the same view and point in time as the encoding frame (Figure 7.1a),

the two should be very similar. Therefore, most blocks should be able to use motion vector of zero and only blocks that are significantly different in the predicted image will actually need to do a motion vector search and encode the motion vector.

P-Stream Reference Mask

Majority of the bits used for encoding reference masks are for primitives *list of boxes* and *list of pixels*. A very simple improvement can be made by using a differential encoding scheme for these two primitives. Using a differential encoding scheme for specifying the boxes and pixels, like the threshold values used for the primitive *delta depth threshold*, should improve encoding of reference masks.

Also, in addition to the *motion compensated box* mentioned in the previous section, another primitive that would be very interesting to examine is a *box with delta depth threshold*. A box with delta depth threshold specifies a reference frame, a threshold value, and a box. For all pixels inside the box, the color is predicted from the specified reference frame only if the difference between the depth value of the encoding pixel and the depth value of the corresponding pixel in the reference frame is less than the given threshold. Using this primitive, regions of reference images can be specified to decrease errors introduced in the initialization step of reference mask encoding.

7.2.2 Encoding Analysis

Another area of future work would be to analyze the various decisions on quality and their effects on multiple depth stream encoding. First is the quality of reference stream encoding and its effects on non-reference stream encoding. Since non-reference streams uses reference streams as basis streams for spatial prediction, reference streams should generally be encoded at higher quality. Determining the relationship between quality of reference stream encoding and non-reference stream encoding should help establish a criterion for encoding multiple depth streams.

Another is the quality of depth encoding and its effect on P-Stream color encoding. As shown in Section 4.3, as the quality of depth improves, P-Stream color encoding size decreases due to superior spatially predicted reference colors. This is because depth is used to spatially

predict color reference values. However, increasing depth quality also increases the depth encoding size. Identifying the relationship between depth and P-Stream color should help in designing an overall guideline for encoding multiple depth streams.

Lastly is to analyze the quality of reference mask encoding on color encoding for P-Streams. The quality of reference mask encoding determines the predicted reference values used for encoding. Generally, a higher quality reference mask predict superior reference values which reduces residual for encoding. This should in turn improve encoding efficiency. However, the error correction step in reference mask encoding is relatively expensive, and the benefits diminish as the number of error corrected pixels decrease. Establishing a general principle for terminating error correction would be beneficial.

7.2.3 Further Evaluation

Finally, as with all evaluations, further assessment of the encoding algorithm on wider set of data would be very useful. All evaluation of the multiple depth stream encoding algorithm presented in this dissertation was done on two data sets – Breakdancers and Ballet from [Zitnick et al. 2004]. Even though the two data sets represents a fast motion scene (Breakdancers) and a slow motion scene (Ballet), the spatial range of motion for both data sets was relatively limited – considerable regions of the frame showed the background scene throughout the sequence. Evaluating the multiple depth stream encoding algorithm on a scene with more moving objects would be an interesting future work.

BIBLIOGRAPHY

[Baker et al. 2002] Baker, H. H., Tanguaya, D., Sobel, I., Gelb, D., Goss, M. E., Culbertson, W. B., and Malzbender, T. (2002). The Coliseum immersive teleconferencing system. In *Proceedings of International Workshop on Immersive Telepresence 2002*, pages 5–8, Juan Les Pins, France.

[Canesta] Canesta, Inc. CanestaVision^{TM}. `http://www.canesta.com`.

[Chai et al. 2000] Chai, J.-X., Chan, S.-C., Shum, H.-Y., and Tong, X. (2000). Plenoptic sampling. In *Proceedings of 27th the annual Conference on Computer Graphics and Interactive Techniques*, pages 307–318, New Orleans, LA, USA.

[Chan et al. 2003] Chan, S.-C., Ng, K.-T., Gan, Z.-F., Chan, K.-L., and Shum, H.-Y. (2003). The compression of simplified dynamic light fields. In *Proceedings of EEE International Conference on Acoustics, Speech, and Signal Processing 2003*, pages III–653–656.

[Chang et al. 1999] Chang, C.-F., Bishop, G., and Lastra, A. (1999). LDI tree: A hierarchical representation for image-based rendering. In *Proceedings of 26th the annual Conference on Computer Graphics and Interactive Techniques*, pages 291–298, Los Angeles, CA, USA.

[Fehn et al. 2002] Fehn, C., Kauff, P., Op de Beeck, M., Ernst, F., IJsselsteijn, W., Pollefeys, M., Van Gool, L., Ofek, E., and Sexton, I. (2002). An evolutionary and optimised approach on 3D-TV. In *Proceedings of International Broadcast Conference*, pages 357–365, Amsterdam, The Netherlands.

[Gross et al. 2003] Gross, M., Würmlin, S., Naef, M., Lamboray, E., Spagno, C., Kunz, A., Koller-Meier, E., Svoboda, T., Van Gool, L., Lang, S., Strehlke, K., Moere, A. V., and Staadt, O. (2003). blue-c: A spatially immersive display and 3D video portal for telepresence. *ACM Transactions on Graphics, Special issue: Proceedings of ACM SIGGRAPH 2003*, 22(3):819–827.

[Haskell et al. 1996] Haskell, B. G., Puri, A., and Netravali, A. N. (1996). *Digital Video: An Introduction to MPEG-2*.

[ITU-T 1995] ITU-T (1995). *ITU-T Recommendation H.263 : Video Coding for Low Bit Rate Communication*.

[Jain et al. 1999] Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323.

[JPEG 1994] JPEG (1994). *Information Technology – Digital Compression and Coding of Continuous-Tone Still Images: Requirements and Guidelines*. ISO/IEC SC29/WG1 10918-1.

[JPEG 2000] JPEG (2000). *Information Technology – JPEG 2000 Image Coding System: Core Coding System*. ISO/IEC SC29/WG1 15444-1.

[Kanade et al. 1996] Kanade, T., Yoshida, A., Oda, K., Kano, H., and Tanaka, M. (1996). A stereo machine for video-rate dense depth mapping and its new applications. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 196–202, San Francisco, CA, USA.

[Kauff and Schreer 2002] Kauff, P. and Schreer, O. (2002). An immersive 3D video-conferencing system using shared virtual team user environments. In *Proceedings of 4th ACM International Conference on Collaborative Virtual Environments*, pages 105–112, Bonn, Germany.

[Kum and Mayer-Patel 2005] Kum, S.-U. and Mayer-Patel, K. (2005). Real-time multidepth stream compression. *ACM Transactions on Multimedia Computing, Communications and Applications*, 1(2):128–150.

[Kum and Mayer-Patel 2006] Kum, S.-U. and Mayer-Patel, K. (2006). Intra-stream encoding for multiple depth streams. In *Proceedings of the 16th* , pages 62–67, Newprot, RI.

[Kum et al. 2003] Kum, S.-U., Mayer-Patel, K., and Fuchs, H. (2003). Real-time compression for dynamic 3D environments. In *Proceedings of the 11th ACM International Conference on Multimedia*, pages 185–194, Berkeley, CA, USA.

[Lalonde and Fournier 1999] Lalonde, P. and Fournier, A. (1999). Interactive rendering of wavelet projected light fields. In *Proceedings of Conference on Graphics Interface '99*, pages 170–114, Ontario, Canada.

[Levoy and Hanrahan 1996] Levoy, M. and Hanrahan, P. (1996). Light field rendering. In *Proceedings of 23rd the annual Conference on Computer Graphics and Interactive Techniques*, pages 31–42, New Orleans, LA, USA.

[Magnor and Girod 2000] Magnor, M. and Girod, B. (2000). Data compression in image-based rendering. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(3):338–343.

[Matusik et al. 2000] Matusik, W., Buehler, C., Raskar, R., Gortler, S. J., and McMillan, L. (2000). Image-based visual hulls. In *Proceedings of 27th the annual Conference on Computer Graphics and Interactive Techniques*, pages 369–374, New Orleans, LA, USA.

[Matusik and Pfister 2004] Matusik, W. and Pfister, H. (2004). 3D TV: A scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. *ACM Transactions on Graphics, Special issue: Proceedings of ACM SIGGRAPH 2004*, 23(3):814–824.

[McMillan and Bishop 1995] McMillan, L. and Bishop, G. (1995). Plenoptic modeling: An image-based rendering system. In *Proceedings of 22nd the annual Conference on Computer Graphics and Interactive Techniques*, pages 39–46, Los Angeles, CA, USA.

[Miller et al. 1998] Miller, G., Rubin, S., and Poncelen, D. (1998). Lazy decompression of surface light fields for precomputed global illumination. In *Proceedings of 9th Eurographics Workshop on Rendering Techniques*, pages 281–292, Vienna, Austria.

[MPEG 1992] MPEG (1992). *Information Technology – Generic Coding of Moving Pictures and Assoicated Audio Information: Video.* ISO/IEC JTC1/SC29/WG11 13818-2.

[MPEG 2004] MPEG (2004). *Information Technology – Coding of Audio-Visual Objects – Part2: Visual*. ISO/IEC JTC1/SC29/WG11 14496-2.

[MPEG 2005] MPEG (2005). *Information Technology – Coding of Audio-Visual Objects – Part10: Advanced Video Coding*. ISO/IEC JTC1/SC29/WG11 14496-10.

[Mulligan et al. 2002] Mulligan, J., Isler, V., and Daniilidis, K. (2002). Trinocular stereo: A real-time algorithm and its evaluation. *International Journal of Computer Vision*, 47(1-3):51–61.

[Peña et al. 1999] Peña, J. M., Lozano, J. A., and Larrañaga, P. (1999). An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters*, 20(10):1027–1040.

[Peter and Straßer 2001] Peter, I. and Straßer, W. (2001). The wavelet stream: Interactive multi resolution light field rendering. In *Proceedings of 12th Eurographics Workshop on Rendering Techniques*, pages 127–138, London, UK.

[Richardson 2003] Richardson, I. E. G. (2003). *H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia*.

[Schreer et al. 2001] Schreer, O., Brandenburg, N., Askar, S., and Trucco, E. (2001). Real-time disparity analysis for applications in immersive tele-conference scenarios – a comparative study. In *Proceedings of 11th International Conference on Image Analysis and Processing*, pages 346–353, Palermo, Italy.

[Shade et al. 1998] Shade, J., Gortler, S., He, L., and Szeliski, R. (1998). Layered depth images. In *Proceedings of 25th the annual Conference on Computer Graphics and Interactive Techniques*, pages 231–242, Orlando, FL, USA.

[Skodras et al. 2001] Skodras, A. N., Christopoulos, C. A., and Ebrahimi, T. (2001). The JPEG2000 still image compression standard. *IEEE Signal Processing Magazine*, 18(5):36–58.

[Smolic and Kauff 2005] Smolic, A. and Kauff, P. (2005). Interactive 3-D video representation and coding technologies. *Proceedings of the IEEE*, 93(1):98–110.

[Taubman and Marcellin 2002] Taubman, D. S. and Marcellin, M. W. (2002). *JPEG2000: Image Compression Fundamentals, Standards, and Practice*, volume 642 of *The Kluwer International Series in Engineering and Computer Science*, Norwell, Massachusetts.

[Towles et al. 2002] Towles, H., Chen, W.-C., Yang, R., Kum, S.-U., Fuchs, H., Kelshikar, N., Mulligan, J., Daniilidis, K., Holden, L., Zeleznik, B., Sadagic, A., and Lanier, J. (2002). 3D tele-immersion over internet2. In *Proceedings of International Workshop on Immersive Telepresence 2002*, pages 28–31, Juan Les Pins, France.

[Vetro et al. 2006] Vetro, A., Su, Y., Kimata, H., and Smolic, A. (2006). *Joint Draft 1.0 on Multiview Video Coding*. Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T, Hangzhou, China. Document JVT-U209.

[Welch et al. 2005] Welch, G., Sonnenwald, D., Mayer-Patel, K., Yang, R., State, A., Towles, H., Cairns, B., and Fuchs, H. (2005). Remote 3D medical consultation. In *Proceedings of*

1st IEEE/CreateNet International Workshop on Telemedicine Over Broadband and Wireless Networks, Boston, MA, USA.

[Würmlin et al. 2004] Würmlin, S., Lamboray, E., and Gross, M. (2004). 3D video fragments: Dynamic point samples for real-time free-viewpoint video. *Computers & Graphics, Special Issue on Coding, Compression and Streaming Techniques for 3D and Multimedia Data*, 28(1):3–14.

[XviD] XviD. `http://www.xvid.org`.

[Yang and Pollefeys 2003] Yang, R. and Pollefeys, M. (2003). Multi-resolution real-time stereo on commodity graphics hardware. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 211–217, Madison, Wisconsin, USA.

[ZCam] 3DV Systems. Z-Cam^{TM}. `http://www.3dvsystems.com`.

[Zitnick et al. 2004] Zitnick, C. L., Kang, S. B., Uyttendaele, M., Winder, S., and Szeliski, R. (2004). High-quality video view interpolation using a layered representation. *ACM Transactions on Graphics, Special issue: Proceedings of ACM SIGGRAPH 2004*, 23(3):600–608.